HARNESS SCALING FOR GENERALIZABLE AGENTIC REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Agent harnesses have become a central component of modern agentic systems, shaping how models use tools, manage context, and coordinate long-horizon interaction. Realizing the benefits of these harnesses depends on how effectively the underlying model can make use of them, motivating reinforcement learning directly through the harness. However, existing approaches typically train through a single fixed harness, which can lead to specialization and limited transfer across harness configurations. To address this, we introduce **Harness Scaling**, which extends agentic RL from a single harness to a structured space of diverse harness configurations. Concretely, we construct a factorized harness space, select a compact training subset with balanced factorial coverage, and organize cross-harness training to combine global coverage with controlled variation for the same task. Across multiple benchmarks, model scales, and RL algorithms, Harness Scaling consistently improves generalization to unseen white-box configurations. Moreover, models trained through structured white-box variation transfer effectively to unseen external black-box harnesses, outperforming models trained through a single complex black-box harness. These results establish harness diversity as a meaningful scaling dimension for agentic RL and highlight the importance of how this diversity is organized for cross-harness generalization.

1 INTRODUCTION

Agent harnesses are playing an increasingly central role in modern agentic systems (Tang et al., 2026). By supporting sustained interaction, tool use, state maintenance, and recovery from failures (Chen et al., 2026), harnesses have substantially improved agent performance on complex, long-horizon tasks (Yang et al., 2024; Ursekar et al., 2026; Ma et al., 2026). Yet the benefits of a capable harness depend not only on its design, but also on whether the underlying policy can use it effectively (Lin et al., 2026b). This naturally motivates optimizing the policy directly through the harness with reinforcement learning (Xu et al., 2026; Yu et al., 2026), which strengthens its ability to exploit the available interaction mechanisms during task solving.

However, existing approaches typically optimize the policy through fixed harness configurations (He et al., 2026; Wang et al., 2026b). While effective within their training configurations, such optimization may encourage specialization to particular execution configurations and weaken transfer across harnesses. We therefore study **Harness Scaling**, treating harness variation as a structured scaling dimension and extending policy optimization over a diverse space of execution configurations. Unlike task or environment scaling (Song et al., 2026c; Dong et al., 2026; Bai et al., 2026b; Wang et al., 2026c), which expands what agents solve or where they operate, Harness Scaling introduces diversity in how policies execute and interact with the same underlying task and environment, promoting generalization across execution configurations.

A key consideration in Harness Scaling is how to model harness variation and select a compact set of configurations for training. Given the modular nature of agent harnesses (He et al., 2026; Shao et al., 2026), we factorize harness variation along five components: system prompt, tool interface, observation, context management, and workflow. Each component contains multiple realizations, whose combinations define a factorial space of complete harness configurations. As variation along these components increases, the space expands rapidly, while exhaustive training can introduce substantial redundancy under a fixed budget. We therefore preserve low-order factorial structure in a compact training set, balancing individual component realizations and their pairwise combinations through an orthogonal design (Rao, 1947; Hedayat et al., 1999). During reinforcement learning,

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.

the same task is paired with structurally nearby harness configurations, providing controlled local variation within each task while maintaining broad coverage across the training space.

To demonstrate the effectiveness of Harness Scaling, we evaluate cross-harness generalization across multiple benchmarks, model scales, and RL algorithms. By training across diverse harness configurations, Harness Scaling reduces specialization to any single harness and encourages more generalizable agent behavior. Empirically, it consistently outperforms single-harness RL on unseen white-box configurations. More importantly, policies trained through structured white-box harness variation transfer effectively to complex external black-box harnesses (OpenClaw, 2026; Anthropic, 2026), outperforming single-harness black-box training. These results establish harness diversity as a meaningful scaling dimension for agentic RL, while showing that effective scaling depends critically on how this diversity is organized during training. In summary, our contributions are threefold:

- We formulate Harness Scaling as a complementary scaling dimension for agentic reinforcement learning, distinct from conventional scaling over tasks and environments, and study how structured harness diversity affects generalization beyond a single execution setting.
- We develop a structured framework for Harness Scaling that factorizes harness variation, selects a compact, factorially balanced training set, and organizes cross-harness training by assigning structurally nearby configurations to the same task across the training space.
- We demonstrate across benchmarks, model scales, and RL algorithms that Harness Scaling substantially improves generalization to unseen white-box and external black-box harnesses, highlighting both the value of harness scaling and the importance of structured organization.

2 Related Work

2.1 Agentic Reinforcement Learning

Reinforcement learning has emerged as an important paradigm for improving the capabilities of LLMs (Zhao et al., 2023; Shao et al., 2024). Recent work has extended RL from single-turn reasoning to increasingly complex agentic settings, including tool use (Bai et al., 2026a; Song et al., 2025), web navigation (Qi et al., 2025), software engineering (Song et al., 2026b), and professional workflows (Wang et al., 2026a). As agentic RL moves toward broader capability, recent efforts have increasingly scaled training along the *task* and *environment* dimensions. Task scaling (Sun et al., 2025; Bai et al., 2026b; Deng et al., 2026) broadens the distribution of problems used for policy optimization, while environment scaling (Dong et al., 2026; Song et al., 2026c; Wang et al., 2026c) expands the range of external settings in which agents interact and receive feedback. These directions expand what agents learn to solve and where they learn to operate, while often retaining a fixed execution harness. In contrast, we study harness diversity as an additional scaling dimension for agentic RL, focusing on policy learning and generalization across execution configurations.

2.2 Harness Design and Optimization

The design of an agent harness can substantially shape how a language model uses tools, maintains context, and performs long-horizon tasks (Yang et al., 2024; Lin et al., 2026b; Ma et al., 2026). Early agent systems largely relied on manually designed harnesses, while subsequent work has explored automated harness improvement through search, editing, and evolution based on task outcomes or execution trajectories (Lin et al., 2026a; Lee et al., 2026; Shao et al., 2026). A complementary line of work directly optimizes model policies within complex agent harnesses, including reinforcement learning through black-box harnesses (Xu et al., 2026; He et al., 2026; Huang et al., 2026). Existing work largely focuses on improving harnesses or optimizing policy models through complex harnesses, with recent work exploring policy optimization across heterogeneous harnesses (Song et al., 2026a; Yu et al., 2026). Harness Scaling instead formulates harness variation as a combinatorial configuration space and studies policy optimization for generalization to unseen harness configurations.

3 Method

3.1 Problem Formulation

To study Harness Scaling, we first formalize the space of execution configurations and the generalization problem over it. We construct a factorized harness space, define a structural notion of proximity among its configurations, and formulate the generalization objectives that motivate the structured harness selection and multi-harness training procedures as shown in Figure 1.

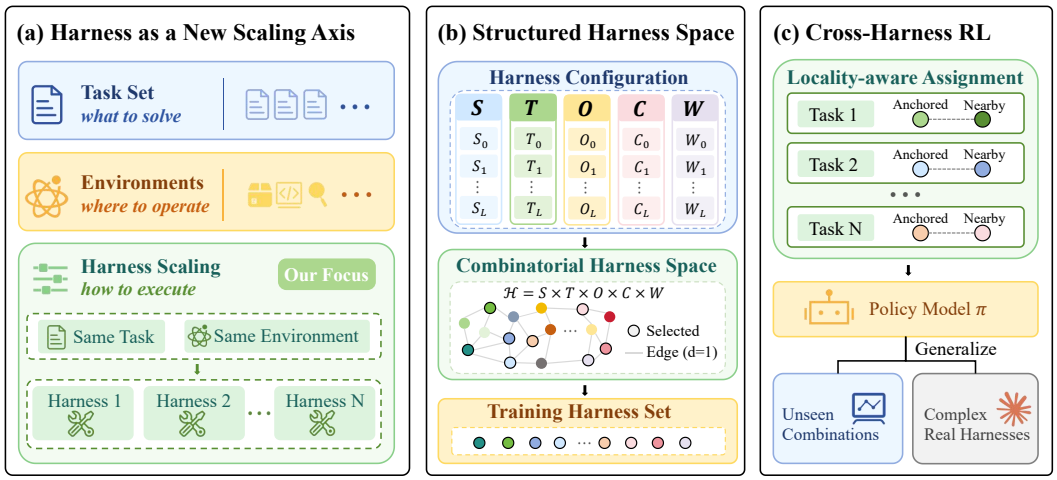


Figure 1: Overview of Harness Scaling, from structured harness-space construction to cross-harness training and generalization to unseen configurations.

3.1.1 Factorized Harness Space

A harness serves as the execution layer through which a policy interacts with its environment. Rather than treating it as monolithic, we factorize the harness into five components: *system prompt*, *tool interface*, *observation*, *context management*, and *workflow*, denoted by S, T, O, C, and W, respectively. These components govern the instructions provided to the policy, tool access, environment feedback, interaction history, and execution procedure, collectively shaping how the policy perceives, acts, and maintains state throughout task execution. This factorization is grounded in common functional components of modern agent harnesses while allowing their effects to interact.

We instantiate each component through a family of component-specific transformations. For a component $F \in \{S, T, O, C, W\}$, let $F^{(0)}$ denote its base realization and $\tau_F^{(\ell)}$ its ℓ-th transformation:

$$F^{(\ell)} = \tau_F^{(\ell)}\left(F^{(0)}\right). \tag{1}$$

A transformation modifies only the target component, keeping the task, environment, and all other harness components fixed. For example, it may reorganize the system prompt, alter tool schemas or naming conventions, change observation feedback, modify context retention or compression, or impose a different execution procedure. Further details are provided in Appendix A.

Let $\mathcal{S}, \mathcal{T}, \mathcal{O}, \mathcal{C}$, and $\mathcal{W}$ denote the resulting realization sets. A complete harness configuration selects one realization from each set, defining the factorized harness space

$$\mathcal{H} = \mathcal{S} \times \mathcal{T} \times \mathcal{O} \times \mathcal{C} \times \mathcal{W}. \tag{2}$$

Each configuration $h \in \mathcal{H}$ therefore specifies a complete execution setting for the same underlying task and environment.

The product structure of $\mathcal{H}$ also induces a natural notion of structural proximity. For $h_i, h_j \in \mathcal{H}$, the Hamming distance $d_{\mathrm{H}}(h_i, h_j)$ counts the number of component dimensions on which they differ. Equivalently, $\mathcal{H}$ forms a Hamming graph whose adjacent configurations differ in exactly one component. We use this distance to organize local cross-harness variation during training.

3.1.2 Generalization Objective

The central objective of Harness Scaling is to learn policies that remain effective beyond the harness configurations observed during training. We consider two complementary forms of cross-harness generalization: broad combinatorial generalization within the constructed harness space and transfer to complex real-world black-box harnesses beyond it.

In-Space Combinatorial Generalization. Let $\mathcal{H}_{\mathrm{tr}} \subset \mathcal{H}$ denote the harness configurations used for training. We evaluate on held-out configurations in $\mathcal{H} \setminus \mathcal{H}_{\mathrm{tr}}$, whose complete combinations are unseen during training. This setting provides a controlled evaluation of unseen combinations within the factorized space.

Out-of-Space Generalization. We further evaluate transfer to complex real-world black-box harnesses whose execution configurations are not represented in $\mathcal{H}$. This setting examines whether training over structured white-box harness variation can transfer to external execution systems beyond the constructed design space.

3.2 Harness Scaling with Structured Selection

As the factorized harness space grows combinatorially with its component dimensions, exhaustive training can become inefficient under a fixed budget. We therefore construct a compact training set that preserves broad factorial coverage under a fixed optimization budget.

3.2.1 Factorial Coverage

The harness space in Eq. 2 can be viewed as a factorial design, where each harness component corresponds to a factor and its realizations correspond to different levels (Rao, 1947; Hedayat et al., 1999). To construct a compact subset without losing basic coverage of this combinatorial space, we preserve its low-order factorial structure: the representation of individual component realizations and their pairwise combinations. These correspond to the first- and second-order projections of the full factorial space, respectively.

Component Balance. We first require the realizations of each component to be represented evenly across the selected configurations. This ensures coverage along every component dimension without overrepresenting particular realizations.

Pairwise Combination Balance. Component balance alone does not control how realizations from different components are combined. We therefore additionally balance all pairwise combinations. Indexing the five component dimensions by $i, j \in \{1, \dots, 5\}$, let $N_{ij}(\ell, \ell')$ denote the number of selected configurations containing the ℓ-th realization of component i together with the ℓ'-th realization of component j. Pairwise balance requires

$$N_{ij}(\ell, \ell') = \lambda, \tag{3}$$

where λ is the common occurrence count for every realization pair of any $i \neq j$. In factorial-design terminology, this corresponds to strength-2 balance, under which every pair of realizations from any two components appears equally often in the selected set.

3.2.2 Harness Set Construction

We construct the training harness set using a strength-2 orthogonal design over the component realizations in $\mathcal{H}$. This directly yields a compact subset satisfying the component-level and pairwise balance defined above, without searching over possible subsets of the full harness space.

The size of such a design is constrained by its pairwise coverage. Suppose there are M component dimensions, each with L realizations, and every pairwise realization combination occurs λ times. Since any two components admit L^2 combinations, the number of selected configurations N satisfies

$$N = \lambda L^2. \tag{4}$$

A strength-2 orthogonal design must satisfy the Rao bound (Rao, 1947; Hedayat et al., 1999),

$$N \geq 1 + M(L - 1). \tag{5}$$

Pairwise balance determines the replication structure of the design, while the Rao bound provides a lower bound on the number of configurations. These conditions restrict the admissible design sizes.

We instantiate a strength-2 orthogonal design satisfying these constraints, yielding the training harness set $\mathcal{H}_{\text{tr}} \subset \mathcal{H}$. The resulting set preserves exact component-level and pairwise balance using only a compact subset of the full factorial space. While these low-order projections are balanced, many complete harness combinations remain unseen during training, enabling evaluation of in-space combinatorial generalization. Further details of the construction and the resulting training configurations are provided in Appendix B.

3.3 Reinforcement Learning across Harnesses

Training each task under multiple harnesses exposes the policy to different execution configurations for the same underlying problem, encouraging the learned behavior to generalize beyond a single harness. We realize this over the selected harness set $\mathcal{H}_{\text{tr}}$ by pairing each task with a small group of structurally related configurations. The assignment is designed to combine controlled variation within each task with broad coverage across the training harness space, and the resulting task–harness instances form the basis for rollout collection and policy optimization.

3.3.1 CROSS-HARNESS TASK ASSIGNMENT

We use a locality-aware strategy to organize how harness configurations are assigned to tasks. For each task x, we assign K configurations from $\mathcal{H}_{\mathrm{tr}}$. An anchor harness $h_x \sim \mathrm{Unif}(\mathcal{H}_{\mathrm{tr}})$ is first sampled uniformly from the training set, and the remaining $K-1$ configurations are chosen from $\mathcal{H}_{\mathrm{tr}} \setminus \{h_x\}$ as those with the smallest Hamming distance $d_{\mathrm{H}}(h_x, h)$ to the anchor. Ties are broken uniformly at random. This associates each task with a small set of nearby configurations while allowing different tasks to cover different regions of the training harness space. The resulting harness set for task x is

$$\mathcal{H}_x = \left\{ h_x,\, h_x^{(1)}, \ldots, h_x^{(K-1)} \right\}, \tag{6}$$

where $h_x^{(1)}, \ldots, h_x^{(K-1)}$ denote the $K-1$ nearest configurations to h_x under d_{H}.

This assignment combines local cross-harness variation with broad coverage over training. For a given task, nearby harnesses provide related but distinct execution configurations, while random anchor selection distributes different tasks throughout $\mathcal{H}_{\mathrm{tr}}$. Each task therefore yields K task–harness instances for rollout collection and policy optimization. When $K = 1$, only the sampled anchor is assigned to the task, while different tasks remain distributed across $\mathcal{H}_{\mathrm{tr}}$.

3.3.2 MULTI-HARNESS POLICY OPTIMIZATION

Each task–harness instance (x, h) is executed under its assigned harness, producing a trajectory τ and the corresponding reward. The K harnesses assigned to the same task are used over K successive training steps, allowing the policy to receive learning signals from different execution configurations for the same underlying task.

For a policy-update batch $\mathcal{B}$ induced by the task–harness training distribution $\mathcal{D}_{\mathrm{MH}}$, let $N_{\mathcal{B}}$ denote the total number of valid policy-generated tokens in the batch. We optimize the standard clipped policy objective:

$$\mathcal{L}(\theta) = \frac{1}{N_{\mathcal{B}}} \sum_{(x,h,\tau)\in\mathcal{B}} \sum_{t=1}^{|\tau|} M_t \min\left(\rho_t(\theta) A_t,\, \mathrm{clip}\big(\rho_t(\theta), 1-\epsilon, 1+\epsilon\big) A_t\right), \tag{7}$$

where $N_{\mathcal{B}} = \sum_{(x,h,\tau)\in\mathcal{B}} \sum_{t=1}^{|\tau|} M_t$, $\rho_t(\theta) = \frac{\pi_\theta(\tau_t | x, h, \tau_{<t})}{\pi_{\theta_{\mathrm{old}}}(\tau_t | x, h, \tau_{<t})}$, and $M_t \in \{0, 1\}$ masks valid policy-generated tokens. The advantage A_t is computed according to the reinforcement learning algorithm.

Harness Scaling determines the task–harness instances used for rollout collection, while the optimization within each step follows the original RL objective. We apply the same cross-harness training procedure to both PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024).

4 EXPERIMENTS

In this section, we first detail the experimental setup and then report the results and detailed analysis.

4.1 EXPERIMENT SETTINGS

Harness Scaling Setup. We instantiate five harness components with three realizations each, yielding $3^5 = 243$ configurations in the full harness space and forming the Hamming graph $H(5, 3)$. Following the strength-2 orthogonal-design construction in Section 3.2.2, the resulting design contains 18 configurations, which form the training harness set $\mathcal{H}_{\mathrm{tr}}$.

Training. We conduct experiments using two backbone models, Qwen3-4B-Instruct and Qwen3-30B-A3B-Instruct (Yang et al., 2025), on ClawGym-Task (Bai et al., 2026b), which contains 13.5K workspace-based general agent tasks. We consider both GRPO (Shao et al., 2024) and PPO (Schulman et al., 2017). For GRPO, each step uses 16 tasks with 8 rollouts per task, while PPO uses 128 tasks with one rollout per task. We set $K = 2$ harnesses per task and train for 300 optimization steps. Additional implementation details and hyperparameters are provided in Appendix C.1.

Evaluation. We evaluate on three benchmarks with varying domains and difficulty levels. ClawGym-Bench (Bai et al., 2026b) covers workspace-based general agent tasks, PinchBench (PinchBench, 2026) focuses on realistic tasks involving research, coding, scheduling, and file operations, and JobBench-Easy (Li et al., 2026) evaluates more challenging professional workflows over heterogeneous workspace files. All evaluation tasks are disjoint from the ClawGym-Task training set.

Table 1: Performance (%) across three benchmarks under different harness evaluation settings. **In-Space** reports the average over 18 held-out harness configurations, while **Claude** denotes out-of-space evaluation with the external black-box harness Claude Code. **Bold** indicates the better result between single-harness training and harness scaling under the same model and optimization method.

Method	Harness	ClawGym-Bench		JobBench-Easy		PinchBench		Avg.
		In-Space	Claude Code	In-Space	Claude Code	In-Space	Claude Code	
Qwen3-4B-Instruct								
Base	–	18.08	33.90	7.16	6.78	32.69	32.74	21.89
GRPO	Single	37.33	38.96	10.24	5.79	45.60	40.95	29.81
	Scaling	**52.78**	**45.37**	**13.55**	**14.83**	**56.24**	**51.47**	**39.04**
PPO	Single	43.78	40.75	12.63	9.68	48.86	47.74	33.91
	Scaling	**49.07**	**44.36**	**13.16**	**13.39**	**53.75**	**47.82**	**36.93**
Qwen3-30B-A3B-Instruct								
Base	–	34.80	37.56	8.44	7.04	49.11	51.48	31.40
GRPO	Single	48.12	46.02	14.79	10.05	54.51	50.25	37.29
	Scaling	**58.60**	**49.62**	**15.48**	**14.20**	**58.98**	**59.78**	**42.78**
PPO	Single	44.23	45.17	13.98	12.11	50.54	51.85	36.31
	Scaling	**48.29**	**48.83**	**17.56**	**18.90**	**57.59**	**60.18**	**41.89**

For in-space generalization, we evaluate on 18 fixed held-out configurations from $\mathcal{H} \setminus \mathcal{H}_{\text{tr}}$, selected for balanced component coverage and diversity across complete harness configurations. The same held-out set is used across all benchmarks, model scales, and RL algorithms. For out-of-space generalization, we use Claude Code (Anthropic, 2026) as the main external black-box harness and additionally evaluate on OpenClaw (OpenClaw, 2026). We report performance averaged over three independent evaluation runs, with in-space scores further averaged across the 18 held-out configurations. Further evaluation details, OpenClaw results, and training-seed robustness are provided in Appendix C.2 and C.3.

Baselines. We compare against four baselines: the original backbone model without additional training; single-harness RL, where the policy is optimized through the fixed base configuration $h_{\text{base}} = (S^{(0)}, T^{(0)}, O^{(0)}, C^{(0)}, W^{(0)})$; black-box SFT, which fine-tunes the policy on trajectories collected through a complex black-box harness; and black-box RL, which optimizes the policy through the same black-box harness. We use OpenClaw as the black-box training harness, while Claude Code is reserved as an unseen external harness for evaluation. All RL baselines are matched in the number of optimization updates and rollouts where applicable.

4.2 Main Results

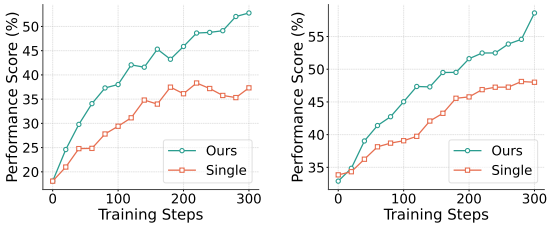
• *Combinatorial Generalization to Unseen Harness Configurations.* Compared with Single-Harness RL, Harness Scaling consistently improves performance on held-out harness configurations unseen during training (Table 1; Figure 2). Across the three benchmarks, average in-space gains are $+9.80$ and $+3.57$ points with GRPO and PPO on Qwen3-4B-Instruct, and $+5.21$ and $+4.90$ points on Qwen3-30B-A3B-Instruct. The same trend holds when Single-Harness RL is trained on three additional fixed harnesses selected solely by their structural positions in $\mathcal{H}_{\text{tr}}$, with detailed results in Appendix C.3. These results support the benefit of Harness Scaling for combinatorial generalization across unseen configurations within the constructed harness space.

• *Consistency across Model Scales and RL Algorithms.* Harness Scaling yields consistent gains across both backbone sizes and optimization methods (Table 1). For Qwen3-4B-Instruct, the overall average improves from 29.81 to 39.04 with GRPO and from 33.91 to 36.93 with PPO. Similar improvements are observed for Qwen3-30B-A3B-Instruct, from 37.29 to 42.78 with GRPO and from 36.31 to 41.89 with PPO. The improvement therefore persists across both model scale and RL algorithm rather than being specific to a single training setting, despite differences in model capacity and optimization dynamics. Additional out-of-space evaluation on OpenClaw shows the same trend in Appendix C.2.

Table 2: Performance (%) on Claude Code as an unseen external black-box harness. Black-Box SFT and Black-Box RL are trained through OpenClaw, while Harness Scaling is trained over the structured white-box harness space. For the RL-based methods, both Black-Box RL and Harness Scaling use GRPO.

Method	ClawGym-Bench	JobBench-Easy	PinchBench	Avg.
Qwen3-4B-Instruct				
Black-Box SFT	44.48	11.10	49.66	35.08
Black-Box RL	44.10	9.42	46.61	33.38
Harness Scaling	**45.37**	**14.83**	**51.47**	**37.22**
Qwen3-30B-A3B-Instruct				
Black-Box SFT	49.55	13.16	51.52	38.08
Black-Box RL	47.84	12.73	48.28	36.28
Harness Scaling	**49.62**	**14.20**	**59.78**	**41.20**

• *Transfer to External Black-Box Harnesses.* Harness Scaling further transfers to the external black-box harness. As shown in Table 2, it achieves average scores of 37.22 and 41.20 with Qwen3-4B-Instruct and Qwen3-30B-A3B-Instruct, respectively, exceeding black-box SFT (35.08 and 38.08) and black-box RL (33.38 and 36.28). Both black-box baselines are trained through OpenClaw and transferred to Claude Code without further adaptation. Notably, Harness Scaling is trained only over the structured white-box harness space, yet still achieves stronger performance on the unseen external harness. This comparison shows that structured training over diverse white-box harnesses can yield stronger cross-harness transfer than direct training through a single complex black-box harness.



(a) Qwen3-4B-Instruct (b) Qwen3-30B-A3B

Figure 2: Evaluation curves under in-space harness configurations on ClawGym-Bench.

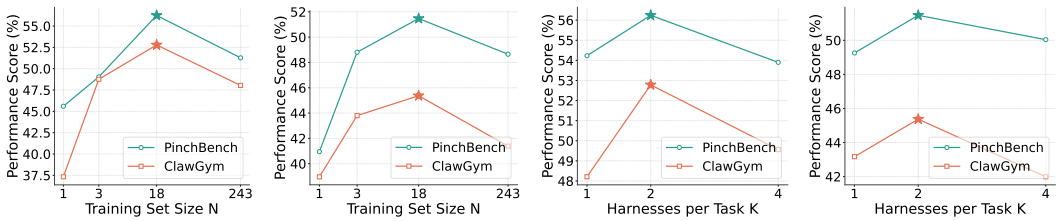
4.3 Ablation Studies

We further examine the key design choices in Harness Scaling, including the construction of the training harness set, the number of harnesses assigned to each task, and the locality-aware assignment strategy. For the training harness set, we compare a single harness, three harnesses providing first-order coverage over all component realizations, our structured 18-harness set, a random 18-harness set, and an exhaustive setting using all 243 configurations. Unless otherwise specified, multi-harness settings use $K = 2$ with the same locality-aware assignment and training budget. The full-space setting includes the white-box evaluation configurations and is therefore used as a fixed-budget stress test rather than a held-out generalization setting. We report the effects of the training harness set and per-task harness diversity in Figure 3. Table 3 further compares our structured 18-harness set with a random set of the same size, and locality-aware assignment with random assignment.

Table 3: Ablation of structured selection and assignment on Qwen3-4B-Instruct. Each entry reports In-Space / Claude Code.

Setting	ClawGym-Bench	PinchBench
Random-18	47.90 / 42.78	51.28 / 48.74
Random Assign.	50.46 / 43.17	53.20 / 50.23
Ours	**52.78 / 45.37**	**56.24 / 51.47**

• *Benefits Depend on Harness Selection.* Moving beyond a single fixed harness substantially improves generalization, but performance does not increase monotonically with the number of training configurations. Using three harnesses with first-order component coverage yields gains of $+11.43/+4.84$ points on ClawGym-Bench and $+3.44/+7.86$ points on PinchBench under in-space/black-box evaluation. Our structured 18-harness set further improves over the three-harness setting by $+4.02/+1.57$ and $+7.20/+2.66$ points, respectively, while also outperforming Random-18 and the full 243-harness setting under the same training budget. This advantage over Random-18 is consistent across three independently sampled harness sets (Table 10) and persists when controlling



(a) White-Box (b) Black-Box (c) White-Box (d) Black-Box

Figure 3: Ablation studies of key design choices in Harness Scaling on Qwen3-4B-Instruct. (a–b) Performance across training harness sets of different sizes under white-box and black-box (Claude Code) evaluation. The full 243-harness setting includes the white-box evaluation configurations and thus serves as a fixed-budget full-space comparison rather than held-out generalization. (c–d) Effect of the number of harnesses per task under the same evaluation settings.

for marginal component balance in Appendix C.3. Notably, the full-space setting is trained on all harness configurations, including those used for white-box evaluation, yet still performs worse than the structured 18-harness set. This suggests that exhaustive coverage alone does not guarantee better performance under a fixed optimization budget, and that how harness diversity is organized matters in addition to configuration count.

• *Effect of Per-Task Harness Diversity.* With $K = 1$, each task uses a single harness, while different tasks remain distributed across the selected harness set. As shown in Figure 3(c–d), this already outperforms Single-Harness RL, showing that cross-task harness diversity is beneficial even without exposing the same task to multiple harnesses. Increasing K from 1 to 2 further improves performance. We additionally evaluate a matched Repeat-2 control that follows the same two-step schedule while reusing the same sampled harness. Repeat-2 shows mixed changes relative to $K = 1$, whereas $K = 2$ consistently performs better across both benchmarks and evaluation settings. This suggests that repeated task exposure alone does not fully explain the gain from $K = 2$, and that cross-harness variation for the same task provides additional benefit. Increasing K to 4 brings no further improvement, with $K = 2$ performing best overall. Detailed results are provided in Appendix C.3.

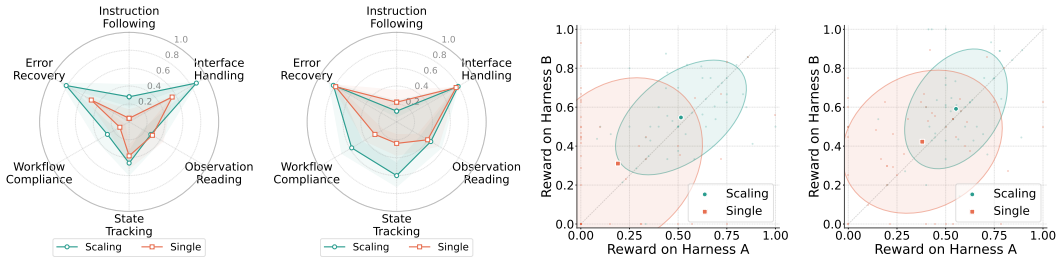
• *Locality-Aware Assignment Complements Global Coverage.* Keeping the structured 18-harness training set fixed, replacing locality-aware assignment with random assignment reduces performance by $2.32/2.20$ points on ClawGym-Bench and $3.04/1.24$ points on PinchBench under in-space/black-box evaluation. This comparison isolates the effect of task–harness assignment while preserving the same global training coverage. The result supports the global–local design of Harness Scaling: random anchor sampling maintains broad coverage across the training harness space, while nearest-harness assignment provides controlled local variation for each task. In contrast, random assignment preserves global coverage but removes this local structure. This suggests that broad global coverage and controlled local variation play complementary roles in Harness Scaling.

5 Analysis

We further investigate behavioral patterns associated with cross-harness generalization. Specifically, we examine whether Harness Scaling is associated with stronger execution capabilities and more robust performance as the execution configuration changes. This analysis complements the task-level results by examining how the learned behaviors differ across harnesses. For each model scale and approach, we analyze 200 complete trajectories per evaluated harness, generated from the same set of ClawGym-Bench tasks to enable direct comparisons across approaches and harnesses. More details are provided in Appendix D.

5.1 Harness-Related Capability Analysis

Harness Scaling strengthens capabilities that are directly relevant to operating under different harness configurations. Figures 4a and 4b compare Single-Harness RL and Harness Scaling along six behavioral dimensions: instruction following, interface handling, observation reading, state tracking, workflow compliance, and error recovery. Across both model scales, Harness Scaling shows consistent improvements over Single-Harness RL on most dimensions.



(a) Qwen3-4B (b) Qwen3-30B-A3B (c) Qwen3-4B (d) Qwen3-30B-A3B

Figure 4: Behavioral analysis comparing Single-Harness RL and Harness Scaling. **(a–b)** Capability profiles under in-space evaluation. **(c–d)** Cross-harness reward consistency on the same tasks under two fixed held-out harnesses, Harness A (`S0T2O1C1W2`) and Harness B (`S2T1O1C2W0`); ellipses show 1.5σ covariance regions and large markers denote group means.

The largest gains appear in state tracking, workflow compliance, and error recovery, indicating better maintenance of task progress, adherence to execution procedures, and recovery from failures. These dimensions are particularly important when changes in the harness alter how context is maintained and how execution is organized. Instruction following and interface handling also improve, while the gain in observation reading is relatively smaller. These results suggest that training across harnesses strengthens capabilities needed to sustain effective execution as the surrounding harness changes.

5.2 Cross-Harness Robustness and Specialization

Harness Scaling yields stronger and more consistent performance across execution configurations. Figures 4(c–d) compare paired rewards for the same tasks under two held-out in-space harnesses: Harness A (`S0T2O1C1W2`) and Harness B (`S2T1O1C2W0`). The two configurations differ in four harness components ($d_{\mathrm{H}} = 4$) and are used consistently for both model scales and training methods. Each point represents one task, and the diagonal denotes equal reward under the two configurations. Under Harness Scaling, the reward distributions shift toward the upper-right region for both model scales, indicating stronger performance under both harnesses. The paired rewards also appear more closely aligned across configurations, suggesting greater consistency as the harness changes.

We further examine whether cross-harness generalization comes at the cost of performance on the base training harness. As shown in Table 4, Harness Scaling incurs only a 0.19-point decrease on the base harness, while improving the ClawGym-Bench in-space score by 10.48 points and Claude Code performance by 3.60 points. The large cross-harness gains relative to the negligible change on the base harness suggest that Harness Scaling improves robustness to execution variation while largely preserving performance under the original training configuration.

Table 4: Base- vs. cross-harness performance on ClawGym-Bench with Qwen3-30B-A3B-Instruct and GRPO. Δ denotes Scaling $-$ Single.

	Base Harness	**In-Space**	**Claude Code**
Single	59.82	48.12	46.02
Scaling	59.63	58.60	49.62
Δ	-0.19	$+10.48$	$+3.60$

6 Conclusion

In this work, we study Harness Scaling as a new scaling dimension for agentic reinforcement learning, motivated by the limitations of optimizing policies through a single fixed execution harness. We formalize harness diversity as a structured combinatorial configuration space and develop a unified framework that integrates structured harness selection with cross-harness policy training. Across benchmarks, model scales, and RL algorithms, Harness Scaling substantially improves generalization to unseen white-box configurations and transfers effectively to complex external black-box harnesses. Our results further show that the benefits of scaling depend on how harness diversity is selected, organized, and incorporated into training, rather than on diversity alone. More broadly, these findings expand the scope of agentic RL scaling beyond what agents solve and where they operate to how they interact and execute, positioning harness variation as a complementary dimension for improving generalization across diverse execution systems.

AI USE STATEMENT

Generative AI tools were used in several stages of this work. As part of the experimental methodology, we used LLMs to instantiate harness component variants from predefined transformation specifications and for model-based evaluation where explicitly described in the paper. Generative AI tools were also used to provide feedback on research methodology and experimental design, and to assist with the analysis and interpretation of experimental results.

In addition, we used generative AI tools to assist with code editing and to improve the clarity and language of the manuscript. All AI-assisted research decisions, experimental implementations, analyses, and manuscript content were reviewed and verified by the authors. Experimental results and quantitative claims were checked against the underlying implementations and outputs. The authors take full responsibility for the final content of this work, including all text, claims, code, analyses, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide detailed information to support reproducibility throughout the paper and appendix. The construction and realizations of the harness space and the structured training harness set are described in Appendix A and Appendix B. Training configurations, hyperparameters, context-compaction procedures, black-box baselines, held-out harness configurations, and evaluation protocols are documented in Appendix C. Additional ablation settings and details of the trajectory-level behavioral analysis are also provided in the appendix.

REFERENCES

Anthropic. Claude Code: Ai-powered coding assistant for developers. `https://claude.com/product/claude-code`, 2026. Product page. Accessed: 2026-04-29.

Fei Bai, Yingqian Min, Beichen Zhang, Zhipeng Chen, Xin Zhao, Lei Fang, Zheng Liu, Zhongyuan Wang, and Hongteng Xu. Towards effective code-integrated reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 30022–30030, 2026a.

Fei Bai, Huatong Song, Shuang Sun, Daixuan Cheng, Yike Yang, Chuan Hao, Renyuan Li, Feng Chang, Yuan Wei, Ran Tao, et al. Clawgym: A scalable framework for building effective claw agents. *arXiv preprint arXiv:2604.26904*, 2026b.

Mengzhuo Chen, Junjie Wang, Zhe Liu, Yawen Wang, Haiming Zheng, and Qing Wang. From failed trajectories to reliable llm agents: Diagnosing and repairing harness flaws, 2026. URL `https://arxiv.org/abs/2606.06324`.

Jia Deng, Yimeng Chen, Xiaoqing Xiang, Ziyang Zeng, Shuo Tang, Wayne Xin Zhao, Feng Chang, Chuan Hao, Yuan Wei, Ran Tao, et al. Fort-searcher: Synthesizing shortcut-resistant search tasks for training deep search agents. *arXiv preprint arXiv:2606.12087*, 2026.

Guanting Dong, Junting Lu, Junjie Huang, Wanjun Zhong, Longxiang Liu, Shijue Huang, Zhenyu Li, Yang Zhao, Xiaoshuai Song, Xiaoxi Li, Jiajie Jin, Yutao Zhu, Hanbin Wang, Fangyu Lei, Qinyu Luo, Mingyang Chen, Zehui Chen, Jiazhan Feng, Ji-Rong Wen, and Zhicheng Dou. Agent-world: Scaling real-world environment synthesis for evolving general agent intelligence, 2026. URL `https://arxiv.org/abs/2604.18292`.

Zhiyuan He, Siwei Zhang, Zhiwen Zhou, Yuqing Yang, Yu Kang, Yuge Zhang, Luna K. Qiu, Tin Yan Tsui, Jiahang Xu, and Chong Luo. Agent lightning v1.0: Towards harnessed agentic rl, 2026. URL `https://arxiv.org/abs/2608.17528`.

A Samad Hedayat, Neil James Alexander Sloane, and John Stufken. *Orthogonal arrays: theory and applications*, volume 68. Springer New York, 1999.

Bo Huang, Fengxiang Li, Hao Xu, Haoyang Huang, Hongyi Fu, Jinhua Hao, Kun Yuan, Minglei Zhang, Pengcheng Xu, Shiyang Liu, Wenhao Zhuang, Yuze Shi, Zongxian Feng, Chao Wang, Cheng He, Chongling Rao, Deyu Cao, Fan Yang, Gang Xiong, Haochen Liu, Jiabao Li, Jian Liang, Jinghui Jia, Jingwen Chang, Jun Du, Junyu Shi, Min Li, Mingqi Wu, Qiang Gao, Shangpeng Yan, Shaotong Qi, Shu Xu, Shuo Zhou, Tiankuo Xu, Tong Zheng, Weilun Zhao, Xiancheng Meng, Xianda Sun, Xiaoyu Jiang, Xunhao Jia, Yao Xia, Yimeng Xu, Yinghan Cui, Yingpeng Chen, Yiwen

Ning, Yong Wang, Yuxuan Sun, Zhongsheng Liu, Ming Sun, Cheng Luo, Chen Yang, Han Li, and Kun Gai. Kat-coder-v2.5 technical report, 2026. URL https://arxiv.org/abs/2607.05471.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023. URL https://arxiv.org/abs/2309.06180.

Hyunin Lee, Jinglue Xu, Jeffrey Seely, Donghyun Lee, Matei Zaharia, and Yujin Tang. Recursive harness self-improvement, 2026. URL https://arxiv.org/abs/2607.15524.

Yuetai Li, Yichen Feng, Zhangchen Xu, Zixian Ma, Kaiyuan Zheng, Fengqing Jiang, Xinghua Sun, Rulin Shao, Zichen Chen, Yue Huang, Xinyang Han, Brian Lee, Kayla Xu, Shenglai Zeng, Hang Hua, Xiangliang Zhang, Basel Alomair, Ranjay Krishna, Luke Zettlemoyer, Pang Wei Koh, Bhaskar Ramasubramanian, Luyao Niu, Xiang Yue, and Radha Poovendran. Jobbench: Aligning agent work with human will, 2026. URL https://arxiv.org/abs/2605.26329.

Jiahang Lin, Shichun Liu, Chengjun Pan, Lizhi Lin, Shihan Dou, Zhiheng Xi, Xuanjing Huang, Hang Yan, Zhenhua Han, Tao Gui, and Yu-Gang Jiang. Agentic harness engineering: Observability-driven automatic evolution of coding-agent harnesses, 2026a. URL https://arxiv.org/abs/2604.25850.

Minhua Lin, Juncheng Wu, Zijun Wang, Zhan Shi, Yisi Sang, Bing He, Zewen Liu, Tianxin Wei, Zongyu Wu, Zhiwei Zhang, Dakuo Wang, Xiang Zhang, Benoit Dumoulin, Cihang Xie, Yuyin Zhou, Suhang Wang, and Hanqing Lu. Harness updating is not harness benefit: Disentangling evolution capabilities in self-evolving llm agents, 2026b. URL https://arxiv.org/abs/2605.30621.

Ziyu Ma, Hailang Huang, Shun Zou, Yong Wang, Shidong Yang, Yiming Hu, Fei Wei, and XiangXiang Chu. Longhorizon-harness: Advancing long-horizon agents for real-world tasks, 2026. URL https://arxiv.org/abs/2608.01964.

OpenClaw. OpenClaw: Personal ai assistant. https://github.com/openclaw/openclaw, 2026. GitHub repository. Accessed: 2026-04-29.

PinchBench. PinchBench: Real-world benchmarks for ai coding agents. https://github.com/pinchbench/skill, 2026. GitHub repository. Accessed: 2026-04-29.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, Tianjie Zhang, Wei Xu, Jie Tang, and Yuxiao Dong. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning, 2025. URL https://arxiv.org/abs/2411.02337.

C Radhakrishna Rao. Factorial experiments derivable from combinatorial arrangements of arrays. *Supplement to the Journal of the Royal Statistical Society*, 9(1):128–139, 1947.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shuai Shao, Kangning Zhang, Qingyao Li, Shijian Wang, Hao Wang, Wenxiang Jiao, Yuan Lu, Yi Guo, Weiwen Liu, and Weinan Zhang. Harness-r1: Learning to edit executable runtime harnesses from agent failure trajectories, 2026. URL https://arxiv.org/abs/2608.02276.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, pp. 1279–1297. ACM, March 2025. doi: 10.1145/3689031.3696075. URL http://dx.doi.org/10.1145/3689031.3696075.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025.

Huatong Song, Fei Bai, Ming Yang, Renyuan Li, Jia Deng, Jujie He, Zhange Zhang, Daixuan Cheng, Yan Xing, Qi Yun, Xuxing Chen, Danyang Li, Feng Chang, Chuan Hao, Ran Tao, Jian Yang, Bryan Dai, Wayne Xin Zhao, Mingjie Tang, and Ji-Rong Wen. Clawgym ii: Exploring black-box rl on agent harness, 2026a. URL https://arxiv.org/abs/2608.16798.

Huatong Song, Lisheng Huang, Shuang Sun, Jinhao Jiang, Ran Le, Daixuan Cheng, Guoxin Chen, Yiwen Hu, Zongchao Chen, Yiming Jia, et al. Swe-master: Unleashing the potential of software engineering agents via post-training. *arXiv preprint arXiv:2602.03411*, 2026b.

Xiaoshuai Song, Haofei Chang, Guanting Dong, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Envscaler: Scaling tool-interactive environments for llm agent via programmatic synthesis, 2026c. URL https://arxiv.org/abs/2601.05808.

Shuang Sun, Huatong Song, Yuhao Wang, Ruiyang Ren, Jinhao Jiang, Junjie Zhang, Fei Bai, Jia Deng, Wayne Xin Zhao, Zheng Liu, et al. Simpledeepsearcher: Deep information seeking via web-powered reasoning trajectory synthesis. *arXiv preprint arXiv:2505.16834*, 2025.

Xinyu Tang, Han Peng, Guoxin Chen, Yuze Shi, Zitao Su, Peiyu Liu, Wayne Xin Zhao, Yawen Li, and Zhe Xue. Agent systems with harness engineering, 2026. URL https://openreview.net/pdf?id=nM5tDHrQsx.

Varun Ursekar, Apaar Shanker, Veronica Chatrath, Yuan Xue, and Samuel Marc Denton. Vero: A harness for agents to optimize agents, 2026. URL https://arxiv.org/abs/2602.22480.

Bowen Wang, Dunjie Lu, Junli Wang, Tianyi Bai, Shixuan Liu, Zhipeng Zhang, Haiquan Wang, Hao Hu, Tianbao Xie, Shuai Bai, Dayiheng Liu, Que Shen, Junyang Lin, and Tao Yu. Cua-gym: Scaling verifiable training environments and tasks for computer-use agents, 2026a. URL https://arxiv.org/abs/2605.25624.

Yinjie Wang, Xuyang Chen, Xiaolong Jin, Mengdi Wang, and Ling Yang. Openclaw-rl: Train any agent simply by talking. *arXiv preprint arXiv:2603.10165*, 2026b.

Zhaoyang Wang, Canwen Xu, Boyi Liu, Yite Wang, Siwei Han, Zhewei Yao, Huaxiu Yao, and Yuxiong He. Agent world model: Infinity synthetic environments for agentic reinforcement learning, 2026c. URL https://arxiv.org/abs/2602.10090.

Binfeng Xu, Hao Zhang, Shaokun Zhang, Songyang Han, Mingjie Liu, Jian Hu, Shizhe Diao, Zhenghui Jin, Yunheng Zou, Michael Demoret, et al. Polar: Agentic rl on any harness at scale. *arXiv preprint arXiv:2605.24220*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering, 2024. URL https://arxiv.org/abs/2405.15793.

Xiao Yu, Baolin Peng, Ruize Xu, Hao Zou, Qianhui Wu, Hao Cheng, Wenlin Yao, Nikhil Singh, Zhou Yu, and Jianfeng Gao. Openforgerl: Train harness-native agents in any environment. *arXiv preprint arXiv:2607.21557*, 2026.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2), 2023.

Table 5: Harness component realizations instantiated by the model-driven transformation procedure for the $3^5 = 243$ factorized harness space.

Dim.	Aspect	Base	Variant I	Variant II
S	System prompt	Direct system instruction specifying the agent role and execution rules.	The same requirements are embedded in a simulated product context, with additional platform and UI descriptions.	The requirements are reframed through a narrative and role-based setting.
T	Tool interface	Original tool names and schemas, such as `execute_bash` and `file_editor`, without distractors.	Functional tools are exposed through semantically related aliases, together with additional non-functional distractor tools.	Functional tools are exposed through opaque identifiers and augmented with distractor tools, largely removing lexical cues to tool functionality.
O	Observation	Raw tool outputs without additional prefixes or suffixes.	Outputs are augmented with simulated infrastructure metadata, including pod names, namespaces, and scheduling information.	Outputs are augmented with structured XML-style metadata, including tool name, exit code, process ID, duration, and execution node.
C	Context management	The complete interaction history remains available in the active context.	LLM-based compaction periodically summarizes earlier history while retaining the five most recent turns in context.	A more aggressive compaction policy retains the three most recent turns, with summarized history externalized to an auxiliary file.
W	Workflow	Standard free-form execution without additional procedural constraints.	A plan-first workflow requires explicit planning before task execution.	Planning is combined with periodic progress verification every three steps.

A Harness Component Variants

We instantiate the factorized harness space through a model-driven transformation procedure. Starting from the base harness, Opus-4.7 generates alternative realizations for each of the five components: system prompt (S), tool interface (T), observation (O), context management (C), and workflow (W).

Each transformation modifies only the designated harness component. Across variants, the underlying task instance and environment are kept fixed, while all non-target harness components remain unchanged. The resulting variation therefore reflects changes in how the same task is executed through the harness, rather than changes to the task or environment itself.

This procedure yields three realizations per component and $3^5 = 243$ complete harness configurations. Table 5 summarizes the resulting variants.

System Prompt (S). This dimension varies how the same operational requirements are presented to the policy. The transformed realizations range from direct instructions to product-oriented and narrative formulations while preserving the underlying execution requirements.

Tool Interface (T). The base interface provides five functional tool families: file reading/writing/editing, Bash execution, code interpretation, search, and web fetching. The transformation varies how these tools are exposed while preserving their underlying functionality. Semantic aliases retain partial cues to tool usage, whereas opaque identifiers substantially reduce such cues; distractor tools further increase interface complexity.

Observation (O). This dimension varies the representation of tool feedback. Simulated infrastructure or structured execution metadata is introduced around the original output, while the task-relevant tool result remains unchanged.

Context Management (C). This dimension varies how interaction history remains accessible during long-horizon execution, ranging from full retention to in-context summarization and more

aggressive externalized compaction. Further details on how compaction participates in policy training are provided in Appendix C.

Workflow (W). This dimension varies the procedural structure of task execution, ranging from free-form interaction to explicit planning and periodic progress verification while preserving the underlying task and available actions.

B Additional Details on Structured Harness Selection

B.1 Orthogonal Design and Coverage

We provide additional details on the structured harness selection in Section 3.2.2. Recall that the harness space contains M component dimensions, each with L realizations. We construct the training harness set using a strength-2 orthogonal design, under which every realization pair from any two component dimensions occurs equally often.

Proposition 1 (Coverage of a Strength-2 Orthogonal Design)**.** *Consider a strength-2 orthogonal design over M component dimensions, each with L realizations. If every pairwise realization combination occurs exactly λ times, then every realization of each component appears exactly λL times, and the total number of selected configurations satisfies*

$$N = \lambda L^2. \tag{8}$$

Proof. Consider any two component dimensions. By the strength-2 property, each of their L^2 realization pairs occurs exactly λ times. Fixing one realization of the first component and summing over all L realizations of the second shows that the fixed realization appears exactly λL times. Therefore, every realization within each component is represented equally often.

Moreover, the L^2 realization pairs between any two components partition the selected configurations. Since each pair occurs λ times, the total number of configurations is

$$N = \lambda L^2.$$

□

A strength-2 orthogonal design additionally satisfies the classical Rao bound (Rao, 1947; Hedayat et al., 1999),

$$N \geq 1 + M(L-1). \tag{9}$$

The bound provides a lower bound on the number of configurations required for a strength-2 orthogonal design.

In our harness space, we use $M = 5$ component dimensions with $L = 3$ realizations per component. Proposition 1 requires

$$N = 9\lambda, \tag{10}$$

while the Rao bound gives

$$N \geq 1 + 5(3-1) = 11. \tag{11}$$

Since N must be a multiple of 9, the smallest design size permitted by these constraints is $N = 18$. We instantiate a corresponding strength-2 orthogonal design with $\lambda = 2$. Under this design, each realization of every harness component appears six times, while every realization pair across any two components appears exactly twice.

The resulting 18-harness set provides balanced first-order and pairwise coverage of the full factorial space while leaving many higher-order configurations unseen during training. This provides a compact training set for Harness Scaling while retaining held-out combinations for evaluating in-space combinatorial generalization.

B.2 Training Harness Set

The resulting structured training set consists of the 18 harness configurations listed in Table 6.

Table 6: The 18 configurations in the structured training harness set $\mathcal{H}_{\text{tr}}$.

S0T0O0C0W0	S0T1O1C1W1	S0T2O2C2W2
S1T0O0C1W1	S1T1O1C2W2	S1T2O2C0W0
S2T0O1C0W2	S2T1O2C1W0	S2T2O0C2W1
S0T0O2C2W1	S0T1O0C0W2	S0T2O1C1W0
S1T0O1C2W0	S1T1O2C0W1	S1T2O0C1W2
S2T0O2C1W2	S2T1O0C2W0	S2T2O1C0W1

Table 7: Main training hyperparameters for GRPO and PPO experiments.

Parameter	**4B GRPO**	**4B PPO**	**30B-A3B GRPO**	**30B-A3B PPO**
Prompt batch size	16	128	16	128
Rollouts per prompt	8	1	8	1
Actor mini-batch size	16	128	16	128
Actor learning rate	1×10^{-6}	1×10^{-6}	1×10^{-6}	1×10^{-6}
Max. prompt length	8K	8K	4K	4K
Max. response length	32K	32K	60K	60K
GAE γ/λ	–	1.0 / 1.0	–	1.0 / 1.0
Critic learning rate	–	1×10^{-5}	–	1×10^{-5}
Value clipping	–	0.5	–	1.0
Training GPUs	16	16	32	32

C Implementation Details

C.1 Training

Training Configuration. We train all models with veRL (Sheng et al., 2025) using asynchronous vLLM (Kwon et al., 2023) rollouts on NVIDIA H20 GPUs, using 16 GPUs for Qwen3-4B-Instruct and 32 GPUs for Qwen3-30B-A3B-Instruct. Table 7 summarizes the main training hyperparameters. For GRPO, each update uses 16 tasks with eight rollouts per task, while PPO uses 128 tasks with one rollout per task. We use an actor learning rate of 1×10^{-6}, and no KL or entropy regularization. Dynamic batching is enabled throughout training. For PPO, we use GAE with $\gamma = \lambda = 1$; the critic masks observation tokens from value regression, and its value head is initialized to zero.

Training with Context Compaction. For the context-compaction variants, the policy model itself performs compaction by generating a summary of the earlier interaction history. Context compaction can split a single rollout into multiple training segments. All segments originating from the same rollout share the same rollout-level reward and participate in policy optimization, causing the number of training segments in a batch to vary dynamically. Segment expansion does not alter the rollout-level grouping used by GRPO: for a group of eight original rollouts, reward normalization is always computed over the eight rollout rewards, regardless of how many segments each rollout produces. Segments inherit the group identity and normalized advantage of their parent rollout and do not introduce additional entries when computing the group mean or standard deviation. Policy loss is aggregated with token-level mean reduction over all valid policy-generated tokens in the batch. Each valid token therefore contributes once to the loss regardless of how its parent rollout is segmented, so splitting a rollout into multiple segments does not introduce additional sample-level weight. This preserves both the rollout-level reward semantics and consistent token-level weighting while allowing all valid post-compaction segments to participate in training.

Black-Box Baselines. For the black-box baselines, we use OpenClaw v2026.4.5 as the training harness and reserve Claude Code v2.1.205 as an unseen external harness for evaluation. For both harnesses, we preserve the native execution logic and replace only the model backend with the corresponding evaluated checkpoint. Any harness-specific model I/O adaptation is fixed across checkpoints evaluated under the same harness. Black-Box SFT and Black-Box RL represent two complementary adaptation strategies to the same single black-box harness. Black-Box SFT fine-tunes the policy on 12,800 fixed trajectories collected on ClawGym-Task through OpenClaw for three epochs, while Black-Box RL performs online policy optimization through OpenClaw for 300 updates

Table 8: The 18 held-out configurations used for in-space evaluation.

S0T2O1C1W2	S2T1O1C2W0	S0T1O0C2W2
S2T2O0C0W1	S0T1O2C0W1	S1T0O0C2W1
S0T0O0C1W2	S0T0O1C0W1	S0T2O2C2W0
S1T0O1C1W0	S1T0O2C2W2	S1T1O0C0W0
S1T2O1C0W2	S1T2O2C1W1	S2T0O2C0W0
S2T1O1C1W2	S2T1O2C2W1	S2T2O0C1W0

with 128 rollouts per update. The two baselines are therefore approximately matched in trajectory-level training exposure, with about 38,400 trajectory presentations in each case, although SFT repeatedly reuses a fixed trajectory set whereas RL collects trajectories online. For the comparison in Table 2, Black-Box RL uses GRPO, and Harness Scaling uses the corresponding GRPO-trained checkpoint. Black-Box RL and Harness Scaling use the same backbone models, 300 optimization updates, 128 rollouts per update, the same main GRPO hyperparameters, and the same evaluation protocol. They therefore match the rollout-level training schedule while differing in the harness distribution through which the training rollouts are generated.

C.2 Evaluation

In-Space Aggregation. For compact notation, indices 0, 1, and 2 denote the base realization, Variant I, and Variant II of each harness component, respectively. We evaluate each checkpoint on the 18 held-out harness configurations listed in Table 8. The 18 evaluation configurations are fixed across benchmarks, model scales, and RL algorithms, and are excluded from training in all main experiments and ablations except the full 243-harness setting. In the full-space setting, they are included in training by construction; its white-box result is therefore interpreted as a fixed-budget comparison rather than held-out generalization. The evaluation set provides exact marginal balance across component realizations: for every harness component, each of its three realizations appears exactly six times. The selected configurations also span diverse combinations within the factorized space, with pairwise Hamming distances ranging from 2 to 5. The set is selected solely based on harness configurations and does not use model performance. We compute the benchmark score separately under each of the 18 configurations and average the scores to obtain the reported white-box evaluation performance. For settings in which these configurations are excluded from training, this corresponds to the reported In-Space performance.

Scoring. We report benchmark performance averaged over three independent evaluation runs. For each run, every task is executed once under the corresponding harness setting, and the resulting task-level scores are aggregated following the benchmark evaluation protocol. The reported score is then obtained by averaging the benchmark scores across the three runs. For rubric-based evaluation requiring a model judge, we use GPT-5.4 with the same judging protocol across all methods and harness settings.

Additional Out-of-Space Evaluation. We further evaluate the white-box-trained policies on OpenClaw as an additional external black-box harness. OpenClaw is not included in the factorized harness space or used during the training of Single-Harness RL and Harness Scaling, providing a complementary out-of-space evaluation beyond Claude Code. We report results using the GRPO-trained checkpoints, consistent with the black-box comparison in Table 2. As shown in Table 9, Harness Scaling consistently improves performance over Single-Harness RL across both model scales and all three benchmarks.

C.3 Additional Experimental Details

Sensitivity to Random Harness Selection. For the Random-18 baseline, we uniformly sample 18 configurations from $\mathcal{H}$ after excluding the 18 held-out evaluation configurations. To examine sensitivity to the particular sampled subset, we repeat Random-18 with three independently sampled 18-harness sets under the same training and evaluation configuration. As shown in Table 10, the results are consistent across random harness sets, and our structured 18-harness design outperforms all three sampled Random-18 sets under both in-space and Claude Code evaluation. We additionally consider a marginally balanced Random-18 baseline, where each realization of every harness component appears exactly six times, matching the first-order coverage of our structured design while leaving

Table 9: Out-of-space performance on OpenClaw as an additional external black-box harness. OpenClaw is not used during the training of Single-Harness RL or Harness Scaling. All results use GRPO-trained checkpoints.

Method	ClawGym-Bench	JobBench-Easy	PinchBench	Avg.
Qwen3-4B-Instruct				
Single-Harness RL	43.18	7.67	45.20	32.02
Harness Scaling	**46.84**	**11.25**	**50.11**	**36.07**
Qwen3-30B-A3B-Instruct				
Single-Harness RL	45.31	11.90	52.78	36.66
Harness Scaling	**50.16**	**14.81**	**58.62**	**41.20**

Table 10: Comparison of alternative 18-harness selections on Qwen3-4B-Instruct with GRPO. Random-18 is reported over three independently sampled harness sets, while Marginal-Balanced-18 additionally matches the first-order component coverage of our structured design. Each entry reports In-Space / Claude Code.

Setting	ClawGym-Bench	PinchBench
Random-18 (seed 42)	47.90 / 42.78	51.28 / 48.74
Random-18 (seed 68)	46.90 / 42.16	52.44 / 47.82
Random-18 (seed 98)	48.02 / 43.11	50.73 / 46.95
Marginal-Balanced-18	49.53 / 43.28	52.82 / 46.71
Ours	**52.78 / 45.37**	**56.24 / 51.47**

pairwise combinations unconstrained. Our structured design remains stronger than this control, indicating that marginal balance alone does not explain the improvement.

Per-Task Harness Diversity. Table 11 reports the numerical results for the per-task harness diversity study. We additionally include a Repeat-2 control that matches the two-step task schedule of $K = 2$ while using the same sampled anchor harness for both successive training steps. Relative to $K = 1$, Repeat-2 produces mixed changes across benchmarks, suggesting that repeated exposure to the same task–harness pair is not uniformly beneficial. In contrast, $K = 2$ consistently outperforms Repeat-2 across both benchmarks and evaluation settings, with gains of $3.02/1.77$ points on ClawGym-Bench and $5.90/2.83$ points on PinchBench under in-space/Claude Code evaluation. These results provide additional evidence that cross-harness variation for the same task contributes beyond repeated task exposure alone. Under the fixed training budget, larger K also reduces the number of distinct tasks encountered during training, introducing a trade-off between per-task harness diversity and task coverage.

Sensitivity to the Single-Harness Configuration. Our main Single-Harness RL baseline uses the base harness h_{base}. To examine sensitivity to this choice, we additionally train three fixed-harness baselines selected from $\mathcal{H}_{\text{tr}}$ using only the geometry of the factorized harness space, without reference to evaluation performance. Specifically, we consider $h_{\text{base}} = (0, 0, 0, 0, 0)$, $h_1 = (1, 1, 1, 2, 2)$, $h_2 = (2, 1, 2, 1, 0)$, and $h_3 = (2, 2, 1, 0, 1)$, where the coordinates follow the order (S, T, O, C, W). Among four-harness subsets containing h_{base}, this is one of the maximally separated choices under Hamming distance, with a minimum pairwise distance of four. The selection therefore spans structurally distinct training configurations while remaining independent of model performance. As shown in Table 12, Single-Harness RL varies with the fixed training harness, whereas Harness Scaling remains stronger than all four evaluated single-harness settings under both In-Space and Claude Code evaluation. This provides additional evidence that the observed advantage is not specific to the base training harness.

Robustness Across Training Seeds. To assess sensitivity to stochasticity in reinforcement learning, we repeat the representative Qwen3-4B-Instruct + GRPO setting with three independent training seeds for both Single-Harness RL and Harness Scaling. The same three seeds are used for both methods, while the training setup and evaluation protocol are kept fixed. As shown in Table 13, Harness Scaling consistently outperforms Single-Harness RL across all three training seeds, with mean improvements

Table 11: Effect of per-task harness diversity on Qwen3-4B-Instruct with GRPO. *Repeat-2* matches the two-step task schedule of $K = 2$ while using the same sampled anchor harness for both steps. Each entry reports In-Space / Claude Code performance.

Setting	ClawGym-Bench	PinchBench
Single-Harness RL	37.33 / 38.96	45.60 / 40.95
Harness Scaling, $K = 1$	48.21 / 43.17	54.23 / 49.26
Repeat-2, Same Anchor	49.76 / 43.60	50.34 / 48.64
Harness Scaling, $K = 2$	**52.78 / 45.37**	**56.24 / 51.47**
Harness Scaling, $K = 4$	49.56 / 41.99	53.90 / 50.04

Table 12: Sensitivity of Single-Harness RL to the fixed training harness on Qwen3-4B-Instruct with GRPO. Results are averaged across the three benchmarks. Harness Scaling is included for reference.

Setting	Harness	In-Space	Claude Code
Single-Harness RL	h_{base}	31.06	28.57
Single-Harness RL	h_1	35.79	36.11
Single-Harness RL	h_2	33.52	33.11
Single-Harness RL	h_3	35.15	32.72
Harness Scaling	$\mathcal{H}_{\text{tr}}$	**40.86**	**37.22**

of 15.75 points in-space and 6.79 points on Claude Code. The consistent per-seed gains indicate that the observed improvement is not driven by a particular training seed.

D Additional Analysis Details

D.1 Behavioral Metrics

We use six trajectory-level behavioral metrics to characterize harness-related capabilities beyond task-level performance. **Instruction Following** measures whether the policy correctly follows the system instructions and execution constraints specified by the harness. **Interface Handling** measures whether the policy correctly interprets and uses the available tool interface, particularly when tool names or schemas differ from the base harness. **Observation Reading** measures whether the policy extracts task-relevant information from tool feedback despite additional formatting or distracting metadata. **State Tracking** measures whether the policy maintains an accurate understanding of task progress, intermediate results, and the evolving workspace state throughout long-horizon execution. **Workflow Compliance** measures whether the policy follows procedural requirements imposed by the harness, such as explicit planning or periodic verification. **Error Recovery** measures whether the policy recognizes execution failures, adapts its strategy, and continues toward task completion rather than repeating or propagating the same error.

D.2 Metric Evaluation

We evaluate the six capabilities from complete agent trajectories using GPT-5.4 as the trajectory judge. For each model scale and each of the 18 held-out in-space harnesses, we collect 200 complete trajectories for both Single-Harness RL and Harness Scaling on the same 200 ClawGym-Bench tasks. For each trajectory, the judge is provided with the task instruction, harness specification, and complete interaction history, including policy actions, tool calls, and tool observations.

Each capability is scored independently on a five-point scale. A score of 1 indicates that the corresponding capability is largely absent or repeatedly violated; 2 indicates substantial deficiencies; 3 indicates partial or inconsistent satisfaction; 4 indicates generally correct behavior with minor deviations; and 5 indicates consistent and effective satisfaction of the criterion throughout the trajectory. The judge assigns each dimension separately rather than producing a single overall trajectory score.

We use the same judging prompt, rubric, and judge model across both approaches and all harness settings. For each behavioral dimension, we first average scores over the 200 trajectories within each held-out harness, and then average equally across all 18 harnesses. For visualization, the five-point

Table 13: Robustness across three independent training seeds on Qwen3-4B-Instruct with GRPO on ClawGym-Bench. Each per-seed score is averaged over three independent evaluation runs. The mean and standard deviation in the final row are computed across training seeds.

	In-Space			Claude Code		
Seed	**Single**	**Scaling**	**Δ**	**Single**	**Scaling**	**Δ**
42	37.33	52.78	+15.45	38.96	45.37	+6.41
68	36.89	52.96	+16.07	37.88	44.87	+6.99
98	37.48	53.21	+15.73	38.42	45.40	+6.98
Mean ± Std.	37.23 ± 0.31	52.98 ± 0.22	**+15.75**	38.42 ± 0.54	45.21 ± 0.30	**+6.79**

scores are normalized by dividing by five, resulting in values between 0.2 and 1.0 as reported in Figures 4a and 4b.