WORLD INTELLIGENCE AGENT: UNIFYING WORLD MODELING AND WORLD ACTION FOR DIGITAL AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Digital agents built on large language models have made rapid progress, with world models emerging as their fundamental predictive engines for understanding complex environments. However, current world models are largely fragmented: they lack a unified data pipeline, are often tied to specific domains, and are treated as passive, isolated simulators. We argue that advancing digital agents requires a true *foundation world model*: a universal cognitive engine that not only masters cross-domain simulation but also yields predictive representations robust enough to extend naturally into downstream applications. To realize this vision of **World Intelligence**, we first construct **WIA-Corpus** (5M continual pre-training examples, 9k SFT trajectories, and 134.6k RL prompts) and **WIA-Bench**, the first paired in-domain (1.8k) and out-of-domain (1.7k) evaluation benchmark spanning seven digital domains. Building on this cohesive data foundation, we develop **WIA-WM**, a foundation world model family (4B / 9B / 27B / 35B-A3B), for digital agents, that masters systematic state-transition modeling. WIA-WM achieves the highest average world-model score among all evaluated models on WIA-Bench, surpassing GPT-5.6-sol and Claude-Opus-4.8 by $+3.99\%$ and $+4.34\%$ on the seven-domain OOD suite, and empowers downstream agents via simulation SFT, simulation RL, and inference lookahead. Finally, as a natural extension of this predictive engine, we introduce **WIA-WAM**, which couples prediction and execution via Dual-Perspective Inversion (DPI) and Counterfactual Simulation-Guided Policy Optimization (CSPO). This integration allows the agent to implicitly internalize environment dynamics, moving beyond passive simulation to proactive, robust decision-making. Our dataset, code, and models will be released to facilitate future research.

1 INTRODUCTION

Imagine reaching for a coffee cup: your brain implicitly simulates the ceramic's temperature and weight before you even move. In biological cognition, an internal simulation of the world is the predictive foundation of all intelligence (Friston, 2010; Clark et al., 2013; Bleyenheuft et al., 2009). In the realm of large language model (LLM)-based digital agents, developing a generalized "world model", a system capable of understanding and predicting complex environment dynamics, has become a compelling pursuit (Hafner et al., 2023; Hansen et al., 2024; Zuo et al., 2026; Luo et al., 2026). While recent world models have made substantial progress as forward simulators, they remain largely fragmented: they are often tied to specific environments, lack a unified data foundation, and are treated as passive, standalone artifacts. We argue that the next step for digital agents requires building a true *foundation world model*: a universal cognitive engine that not only masters cross-domain simulation but also possesses predictive representations so robust that they can seamlessly extend to guide downstream applications, including action execution.

This vision motivates a central question: *how can we build and utilize a universal predictive foundation for digital agents?* We call this overarching capability **World Intelligence**. At its cornerstone lies *world modeling*: the ability to maintain a systematic, generative understanding of state transitions across diverse digital environments. Once this robust predictive foundation is established, its capabilities radiate outward. It acts as a powerful simulator for data synthesis and agent training,

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.

and crucially, it naturally unlocks *world action*: the application of its predictive representations to directly optimize an agent's policy. In this paradigm, the foundation world model is the primary pillar of intelligence, and intelligent action generation serves as its ultimate downstream extension.

Achieving World Intelligence requires overcoming three obstacles. The first is data scarcity: public digital-agent corpora are fragmented across the training pipeline (Liu et al., 2024; Yao et al., 2022), with no benchmark jointly evaluating in-domain (ID) and out-of-domain (OOD) generalization, so an end-to-end World Intelligence agent lacks a coherent foundation on which to build. The second is cross-environment world modeling: most existing world models target a single digital interface (Ha & Schmidhuber, 2018; Luo et al., 2026), and their data, architecture, and optimization procedures are tied to that interface, so knowledge does not transfer across environments. The third is world-action decoupling: across existing approaches, world-model predictions are auxiliary signals rather than part of the optimization path (Williams, 1992; Gu et al., 2024), so the gradient of task success never reaches the world model's representations. In policy gradient training, an action that yields a more accurate predicted future state receives no stronger reinforcement than an action that does not, so the policy receives no gradient signal that rewards internalizing the world model.

In this work, we propose the **World Intelligence Agent** (WIA), a unified pipeline that closes all three gaps. We first construct **WIA-Corpus**, spanning 5M CPT examples, 9k SFT trajectories, and 134.6k RL prompts across seven digital domains, and **WIA-Bench**, the first paired ID (1.8k) and OOD (1.7k) evaluation benchmark. Together, they unify training and evaluation across all pipeline stages and environments. Second, from a shared continual-pretrained base **WIA-CPT**, we develop **WIA-WM** (World Model), which establishes systematic state-transition modeling across complex multi-domain environments by learning to simulate how the environment responds to actions. To further improve downstream agents, we explore WIA-WM as a source of simulated fine-tuning, simulated RL, and inference lookahead, yielding accuracy gains on several benchmarks. Third, we introduce **WIA-WAM** (World Action Model), which is initialized from the same WIA-CPT base and couples agent-view and world-view representations through Dual-Perspective Inversion (DPI) and Counterfactual Simulation-Guided Policy Optimization (CSPO), so that improved observation prediction directly shapes more effective actions. Extensive experiments show that WIA achieves strong performance across all benchmarks, with WIA-WM scaling from 4B to 35B-A3B and WIA-WAM demonstrating that directly coupling world modeling with execution yields consistent gains on WebShop, Search-R1, and ScienceWorld.

In summary, our contributions are threefold:

- **WIA-Corpus and WIA-Bench.** We construct a comprehensive corpus of 5M CPT examples, 9k SFT trajectories, and 134.6k RL prompts to address the lack of coherent data for World Intelligence. Together with WIA-Bench, we establish a unified training and evaluation foundation.
- **WIA-WM.** We develop a foundation world model for systematic state-transition modeling across complex multi-domain environments. Trained at multiple scales to study scaling behavior, it empowers downstream agents via simulation SFT, simulation RL, and inference lookahead.
- **WIA-WAM.** We introduce a world action model coupling world modeling and action execution via DPI and CSPO, so that improved observation prediction directly drives effective actions.

2 Related Work

2.1 World Models for Digital Agents

World models predict how environments change in response to actions, enabling agents to evaluate possible consequences without repeated real-environment interaction (Hafner et al., 2023; Hansen et al., 2024; Hao et al., 2023; Lu et al., 2026b; Chae et al., 2025; Zhang et al., 2026). Beyond inference-time planning, several works generate synthetic rollouts for model-based RL (Fang et al., 2025; Ding et al., 2026; Xiao et al., 2026). Most existing world models, however, target a single digital interface, with architectures, data, and training recipes tied to it (Ha & Schmidhuber, 2018; Luo et al., 2026; Tang et al., 2026). In contrast, WIA-WM performs systematic state-transition modeling across multiple digital domains with a unified architecture. We will release our models, code, and training recipes, including the WIA-WM training objectives.

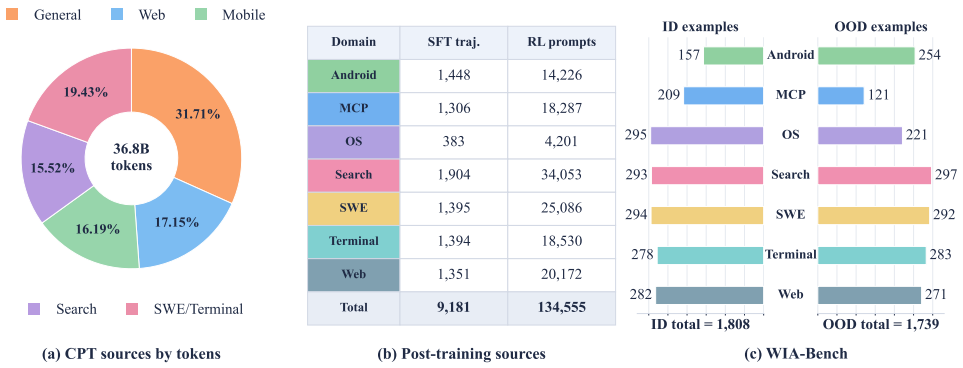


Figure 1: Overview of **WIA-Corpus** and **WIA-Bench**. (a) Token and action–observation supervision distribution of the 36.83B-token continual pre-training corpus. (b) Domain mix of the post-training sources. (c) Final ID and OOD example counts by domain.

2.2 World-Action Models for Digital Agents

Several methods connect world models to policy learning by generating synthetic rollouts, scoring candidates at test time, or performing model-based planning (Guo et al., 2025; Chen et al., 2025a; Yu et al., 2025; Lu et al., 2026a). However, these approaches generally retain the simulator and policy as separate components, leaving their alignment implicit and their deployment paths distinct. WIA-WAM differs by co-training a unified model that learns both action generation and action-conditioned environment dynamics. DPI aligns agent-view and world-view representations, while CSPO propagates task-success gradients through the world-view objective, so that improved prediction directly shapes action generation.

3 Dataset

We construct **WIA-Corpus**, a full-lifecycle training corpus (5.03M continual pre-training examples, 9,181 SFT trajectories, and 134,555 RL samples), together with **WIA-Bench**, a paired in-domain (ID) and out-of-domain (OOD) benchmark spanning seven digital domains: Android, MCP, OS, Search, SWE, Terminal, and Web. All sources are normalized into action–observation trajectories with stage-specific loss masking; Figure 1 summarizes the corpus composition and benchmark splits. Full source-level statistics and construction details are provided in Appendix D.

Continual pre-training sources. We merge five interaction families: general agent trajectories, web observations, mobile UI view trees, research/search sessions, and terminal/SWE rollouts. Each source is cleaned to remove malformed actions, error observations, reasoning traces, noisy content, and overlong observations, and is then unified into multi-turn records with explicit loss masks. Overlong samples are capped, long trajectories are decomposed, and all sources are globally shuffled into balanced shards, yielding 5.03M examples and 36.83B tokens.

Post-training sources. The post-training pool spans the same seven domains: SWE, Search, and MCP are built from agent trajectories self-sampled by our pipelines, while Android, OS, Terminal, and Web additionally draw on public, curated, or synthetic corpora. We reserve a domain-balanced held-out split of 1k trajectories for in-domain evaluation. For SFT, we select 9,181 high-quality trajectories, normalize them at the action-argument level, and rewrite actions into natural-language style with observation-only multi-turn supervision. Turn-level RL examples are filtered and rebalanced, yielding 134,555 prompts with ground-truth observations.

Benchmark construction. ID examples are drawn from the held-out split and converted into the shared first-window, single-turn prediction format. For the OOD suite, six domains are sampled from environments distinct from those used in training, while OS uses recorded desktop-office trajectories. After filtering out unchanged observations, error interfaces, invalid actions, and unreliable samples, WIA-Bench contains 3,547 examples (1,808 ID and 1,739 OOD), with every domain represented in both suites to facilitate a comprehensive evaluation of cross-environment generalization.

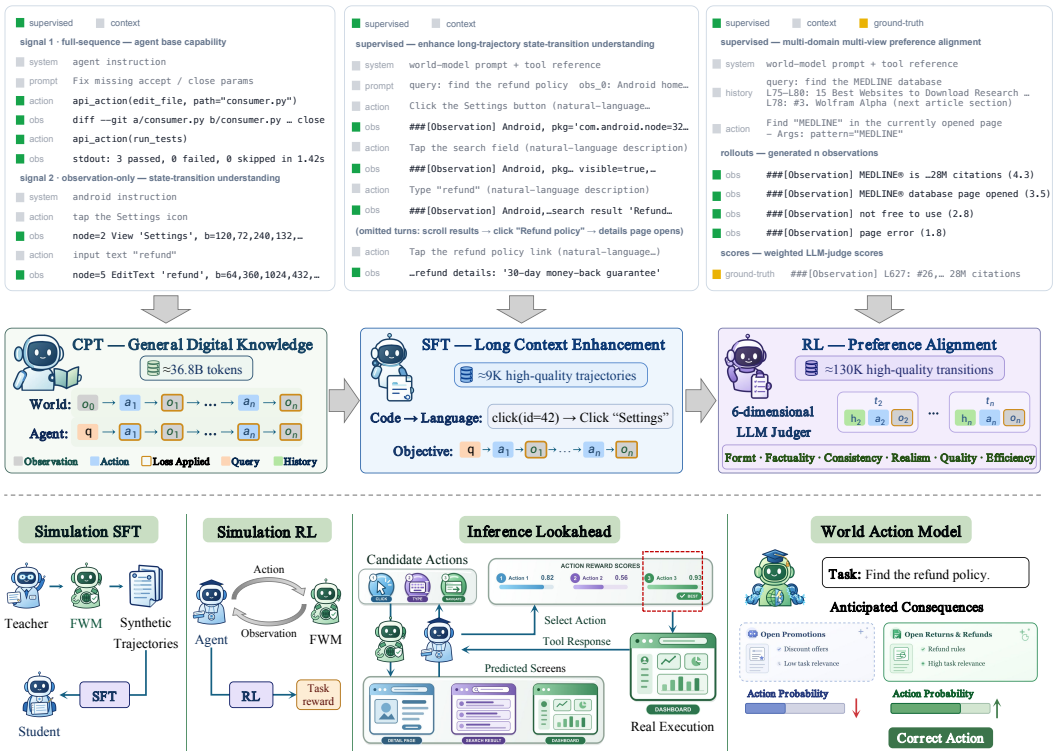


Figure 2: Overview of the WIA training pipeline. Diverse environments are unified into a text-conditioned action–observation interface and trained via three stages (CPT → SFT → RL) to yield WIA-WM, enabling simulated SFT, simulated RL, inference lookahead, and world action modeling.

4 METHOD

Overview. WIA unifies heterogeneous environments into a shared textual action–observation interface (Section 4.1; Figure 2), trains **WIA-WM** through CPT, SFT, and RL for next-observation prediction (Section 4.2), uses it as a simulator (Section 4.3), and trains **WIA-WAM** with DPI and CSPO to predict both actions and observations (Section 4.4).

4.1 CROSS-DOMAIN WORLD MODELING

WIA unifies the seven domains (Android, MCP, OS, Search, SWE, Terminal, and Web) into a shared text-conditioned representation, retaining transition-relevant identifiers, commands, tool arguments, and structured feedback while discarding rendering details. A trajectory proceeds as

$$\tau = (q, o_0, a_1, o_1, \ldots, a_T, o_T, ans), \tag{1}$$

where q is the task instruction, o_0 is an optional initial observation, a_t is an action, o_t is the observation produced by a_t, and ans denotes the agent's final response. Depending on the domain, either q or o_0 is optional; in all cases the only required primitive is predicting o_t from the preceding context and a_t. Domain-specific text serializations, marker schema, and history windows are provided in Appendix D. Concrete per-domain transitions are shown in Appendix H.

4.2 WIA-WM TRAINING PIPELINE

WIA-WM is built progressively through three successive stages that differ in supervision granularity and each address a distinct weakness of the previous stage. Continual pre-training endows the model with broad cross-domain transition priors, supervised fine-tuning grounds these priors in the tool-referenced multi-turn observation format, and reinforcement learning aligns the generated observations with human-preferred world-state quality, yielding reliable next-observation prediction.

Continual pre-training (CPT). A base language model inherently lacks a unified prior over heterogeneous digital-environment transitions. CPT builds this prior at the token level through a **dual-target design**: one shared sequence model is trained on the mixed corpus, where a world target predicts only observations, $Y^{\text{world}} = (o_1, \ldots, o_n)$ over $\tau = (o_0, a_1, o_1, \ldots, a_n, o_n)$, to learn state transitions, while an agent target additionally predicts actions and the final answer, $Y^{\text{agent}} = (a_1, o_1, \ldots, a_n, o_n, ans)$ over $\tau = (q, a_1, o_1, \ldots, a_n, o_n, ans)$, to preserve action-conditioned representations. Supervision is therefore imposed only on the target tokens while the surrounding interface remains visible, which is what lets the same backbone later serve as both a world model and an action model. The objective is the shared masked log-likelihood

$$\mathcal{L}_{\text{sup}}(\theta; \tau, Y) = -\frac{1}{|Y|} \sum_{y_t \in Y} \log p_\theta\big(y_t \mid \tau_{<t}\big), \tag{2}$$

mixed over the two target types by the source proportions in Section 3. This yields **WIA-CPT**, which initializes both WIA-WM and WIA-WAM.

Supervised fine-tuning (SFT). CPT provides broad priors but not the precise tool-grounded multi-turn format. SFT closes this gap through format alignment under observation-only supervision: it injects tool references into the system prompt to explain the action space, rewrites every action as a natural-language description $\tilde{a}_j$, and supervises only observations. Crucially, the action side is rendered but never optimized, so the model learns to ground transitions precisely rather than imitate the demonstration policy. The resulting checkpoint, **WIA-SFT**, reuses $\mathcal{L}_{\text{sup}}$ with $\tau = (q, \tilde{a}_1, o_1, \ldots, \tilde{a}_n, o_n)$ and $Y^{\text{SFT}} = (o_1, \ldots, o_n)$:

$$\mathcal{L}_{\text{SFT}}(\theta) = \mathbb{E}_{\tau \sim \mathcal{D}_{\text{SFT}}} \mathcal{L}_{\text{sup}}\big(\theta; \tau, Y^{\text{SFT}}\big). \tag{3}$$

Reinforcement learning (RL). Likelihood training alone still under-weights human-preferred world-state quality. RL closes this gap through explicit action-level preference alignment: on turn-level prompts (h, a) the model samples G candidate observations and optimizes them directly with GSPO (Zheng et al., 2025) under a length-normalized clipped objective

$$\mathcal{L}_{\text{RL}}(\theta) = -\mathbb{E}_{(h,a)\sim\mathcal{D}_{\text{RL}},\ \{o^{(i)}\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}} \left[\frac{1}{G} \sum_{i=1}^{G} \min\Big(s_i(\theta)\hat{A}_i,\ \text{clip}(s_i(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_i \Big) \right], \tag{4}$$

where $s_i(\theta) = \big(\pi_\theta(o^{(i)} \mid h, a) / \pi_{\theta_{\text{old}}}(o^{(i)} \mid h, a)\big)^{1/|o^{(i)}|}$ and $\hat{A}_i$ is the group-normalized advantage. The rollout reward is produced by a six-dimensional judge over format, factuality, consistency, realism, quality, and efficiency. Each dimension score $s_k \in [1, 5]$ is rescaled into $[0, 1]$ and combined by the weights $\mathbf{w} = (1, 2, 2, 2, 2, 1)$ as $R = \frac{1}{10} \sum_{k=1}^{6} w_k \, (s_k - 1)/4 \in [0, 1]$, which serves as the optimization signal and yields **WIA-WM**. The pseudocode is in Algorithm I.1 (Appendix I).

4.3 World Model as a Simulator

Beyond prediction evaluation, WIA-WM serves as a simulator in three ways (Figure 3).

Simulation SFT. A teacher policy supplies reference actions and WIA-WM predicts the corresponding observations, forming synthetic interaction trajectories on which a student policy is fine-tuned. In this way, high-quality teacher behavior is distilled into diverse, reusable rollouts without collecting additional real-environment samples, which is especially useful when real interactions are costly or cannot cover rare and long-tail actions.

Simulation RL. The agent interacts with WIA-WM: it issues a candidate action, receives a predicted observation, and obtains a task reward, and the policy is optimized on these simulated rollouts. Because the world model can be queried many times, the agent can explore and recover from mistakes without running the real environment; compared with SFT alone, the reward signal enables the agent to correct mistakes and sharpen tool grounding across repeated simulated trials.

Inference lookahead. At each step the policy samples multiple candidate actions and WIA-WM predicts the corresponding next observation for each candidate; a selector compares these predicted outcomes with the real history and then selects the most favorable one, and only that action is actually executed in the real environment. This adds only a small sampling and ranking cost while the environment is updated with a single real observation.

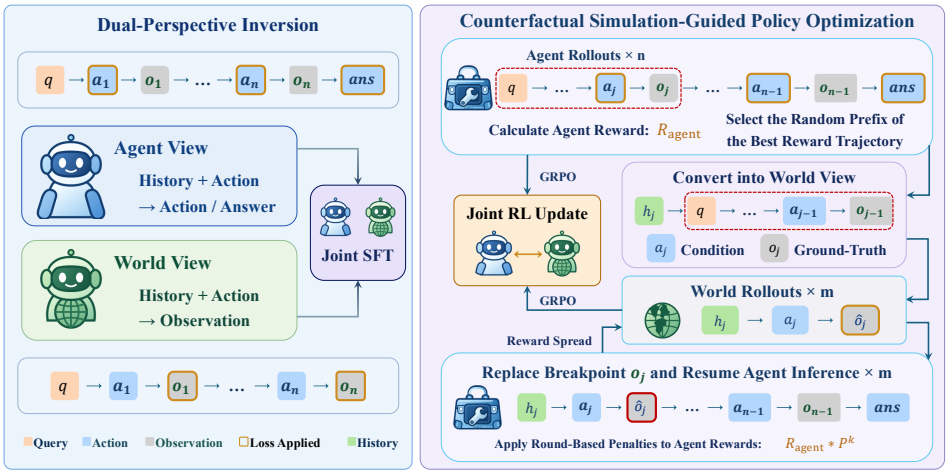


Figure 3: Illustration of WIA-WAM post-training. **DPI (left)** fine-tunes an agent view for action prediction and a world view for observation prediction. **CSPO (right)** samples counterfactual observations and propagates continuation rewards through GRPO to both objectives.

4.4 WIA-WAM Post-Training

WIA-WAM is initialized from **WIA-CPT** and internalizes environment dynamics, so that inference requires no explicit lookahead step. It is trained in two stages (Figure 3): **Dual-Perspective Inversion (DPI)** for supervised fine-tuning and **Counterfactual Simulation-Guided Policy Optimization (CSPO)** for reinforcement learning. DPI learns both agent-view action generation and world-view action-conditioned observation prediction with shared parameters, thereby aligning the two views; CSPO then uses the predicted next observations to reinforce the policy.

For DPI, each expert trajectory is rendered into an agent-view example in $\mathcal{D}_P$ and a world-view example in $\mathcal{D}_W$. The agent view predicts the next action from instruction q and history $H_j = (a_1, o_1, \ldots, a_{j-1}, o_{j-1})$:

$$\mathcal{L}_{\mathrm{PM}}(\theta) = -\mathbb{E}_{(H_j, a_j)\sim\mathcal{D}_P}\left[\log P_\theta(a_j \mid q, H_j)\right], \tag{5}$$

while the world view truncates after action a_j and predicts the resulting observation o_j:

$$\mathcal{L}_{\mathrm{WM}}(\theta) = -\mathbb{E}_{(H_j, a_j, o_j)\sim\mathcal{D}_W}\left[\log P_\theta(o_j \mid q, H_j, a_j)\right]. \tag{6}$$

The two losses are jointly optimized via a shared sequence model: every expert trajectory is rendered twice: once for action prediction and once for observation prediction—forcing one unified prefix to support both tasks without extra demonstrations.

For CSPO, each task first samples N_a policy rollouts $\mathcal{G}_p$ and scores them with the task reward, keeping the highest-reward rollout as the reference. It then uniformly samples a truncation step j ($1 \le j < T$). Given the prefix ending with a_j, the world model samples N_w rollouts $\mathcal{G}_w$, each producing a candidate observation $\hat{o}^{(m)}$; each is inserted as the pseudo-observation for step j, and the policy resumes to obtain continuation reward r_m^{cont}. Extra rounds are penalized:

$$\tilde{r}_m^{\mathrm{cont}} = r_m^{\mathrm{cont}} \cdot 0.95^{\Delta T_m}, \qquad \Delta T_m = \max(0, T_m^{\mathrm{cont}} - T^{\mathrm{ref}}). \tag{7}$$

We compute task rewards for policy rollouts and continuation rewards for the world rollouts, group-normalize each set into advantages A_i^{policy} and A_m^{world}, and combine them in the Group Relative Policy Optimization (GRPO) objective (Shao et al., 2024):

$$\mathcal{L}_{\mathrm{CSPO}} = \lambda_p \mathcal{L}_{\mathrm{GRPO}}\left(\mathcal{G}_p, A^{\mathrm{policy}}\right) + \lambda_w \mathcal{L}_{\mathrm{GRPO}}\left(\mathcal{G}_w, A^{\mathrm{world}}\right), \tag{8}$$

where λ_p and λ_w balance the two objectives. Unlike MuZero/Dreamer, CSPO uses no separate dynamics or reward model—it splices a world-view prediction into real trajectories and treats actual task reward as the world-view advantage ("counterfactual" here means an alternative next observation, not a formal causal-inference claim). Only the policy is deployed. Reward construction and rollout counts are in Algorithm I.2 (Appendix I), with clipping schedules in Appendix A.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

We build **WIA-CPT** on WIA-Corpus from four Qwen3.5 backbones (4B/9B/27B/35B-A3B) (Team, 2026), then train **WIA-WM** via world-model SFT followed by GSPO-based RL, and **WIA-WAM** via DPI SFT followed by CSPO. All stages use BF16 full-parameter training; CPT, SFT, and RL settings are detailed in Appendix A, and data distributions in Appendix D. The WIA-WM RL stage uses `DeepSeek-V4-Flash` (Xu et al., 2026) as the world-model judge, and WIA-Bench is scored by `DeepSeek-V4-Pro` (Xu et al., 2026). We benchmark against proprietary models (GPT-5.6 (OpenAI, 2026), Claude (Anthropic, 2026b;a)) and open-source baselines (GLM-5.3 (Zhipu AI, 2026), MiMo-v2.5-pro (MiMo, 2026), DeepSeek-V4 (Xu et al., 2026), Qwen3.5/3.8 (Team, 2026; Qwen Team, 2026), Qwen-AgentWorld (35B-A3B) (Zuo et al., 2026)). Scores follow the six-dimensional prompt from Section 4.2, rescaled to a 100-point scale; judge prompts are provided in Appendix G.

5.2 MAIN RESULTS

Figure 4 and Table 1 report the world-model ranking and OOD results under the DeepSeek-V4-Pro judge. WIA-WM achieves the highest sample-weighted average on both suites: **66.41** on ID with WIA-WM-27B and **55.72** on OOD with WIA-WM-35B-A3B. Our 27B dense and 35B-A3B MoE models achieve OOD Avg. values of 55.45 and 55.72, ahead of the strongest baselines: GPT-5.6-sol (OpenAI, 2026) (53.58), Claude-Opus-4.8 (Anthropic, 2026a) (53.40), Qwen-AgentWorld (52.87), and Qwen3.8-max (Qwen Team, 2026) (52.40). Across the seven domains, the WIA-WM family leads on SWE (60.77), Search (51.33), and Android (50.51), ranks second on MCP (52.53).

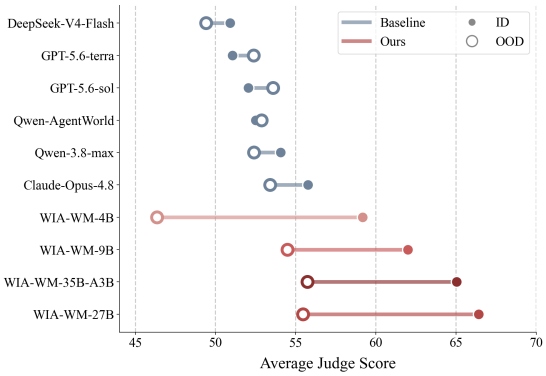


Figure 4: Comparison of performance on ID and OOD benchmarks for the top-10 models.

Table 1: OOD world-model results under the DeepSeek-V4-Pro judge. Red and blue cells indicate the best and second-best results per column; gray rows mark our WIA-WM models. `Avg` is the sample-weighted average across samples, `MAvg` is the macro average across the seven domains.

	Model	Android	MCP	OS	Search	SWE	Terminal	Web	Avg	MAvg
Proprietary	GPT-5.6-luna	48.37	44.51	54.96	40.01	55.88	62.42	53.87	51.91	51.43
	GPT-5.6-terra	48.34	43.55	56.14	40.46	57.68	61.79	54.57	52.38	51.79
	GPT-5.6-sol	**49.39**	45.75	59.70	39.99	58.61	**63.63**	55.00	53.58	53.15
	Claude-Sonnet-5	47.76	49.85	63.22	35.26	44.58	46.65	54.24	48.03	48.79
	Claude-Opus-4.8	48.98	**53.97**	63.02	39.10	58.15	57.93	**55.26**	53.40	53.77
Open-source	GLM-5.3	47.20	50.23	55.58	31.10	32.68	53.88	51.03	44.98	45.96
	MiMo-v2.5-pro	45.80	46.18	59.78	38.20	49.73	53.43	51.50	49.09	49.23
	Qwen3.8-27B	41.08	39.40	53.34	34.88	48.47	51.84	47.51	45.46	45.22
	Qwen3.8-max	47.60	45.32	57.97	36.82	57.62	**63.42**	**55.48**	52.40	52.03
	DeepSeek-V4-Flash	46.74	44.46	58.93	38.40	50.51	55.53	50.77	49.40	49.33
	DeepSeek-V4-Pro	47.87	43.92	61.93	38.77	51.14	57.77	52.63	50.73	50.57
Qwen3.5	Qwen3.5-4B	39.92	38.38	55.29	30.44	37.26	39.50	45.24	40.46	40.86
	Qwen3.5-9B	43.00	41.37	58.19	30.99	44.05	43.81	47.42	43.76	44.12
	Qwen3.5-27B	47.84	42.56	**64.29**	33.75	48.10	55.09	52.78	49.15	49.20
	Qwen3.5-35B-A3B	47.09	42.69	56.59	32.03	45.70	49.80	49.55	46.01	46.21
	Qwen-AgentWorld	47.43	42.63	**64.21**	45.02	56.31	57.22	53.68	52.87	52.36
	WIA-WM-4B	27.19	44.10	48.40	36.84	**60.06**	55.87	49.27	46.34	45.96
	WIA-WM-9B	46.82	52.12	62.70	46.06	**60.77**	59.38	53.43	54.49	54.47
	WIA-WM-27B	**50.51**	**52.53**	63.42	**46.36**	59.92	60.73	54.52	**55.45**	**55.43**
	WIA-WM-35B-A3B	48.76	43.72	63.86	**51.33**	59.19	62.52	54.95	**55.72**	**54.90**

On the OOD suite, WIA-WM improves monotonically with scale, from 46.34 at 4B to 55.72 at 35B-A3B. The 9B model surpasses every OOD baseline at 54.49 despite far fewer parameters. Qwen-AgentWorld shows a similar pattern: over Qwen3.5-27B it lifts SWE from 48.10 to 56.31 and Search from 33.75 to 45.02, and WIA-WM further improves SWE and MCP beyond Qwen-AgentWorld. Additional ID results and robustness checks under the DeepSeek-V4-Flash are in Appendix B.

5.3 Simulation Results

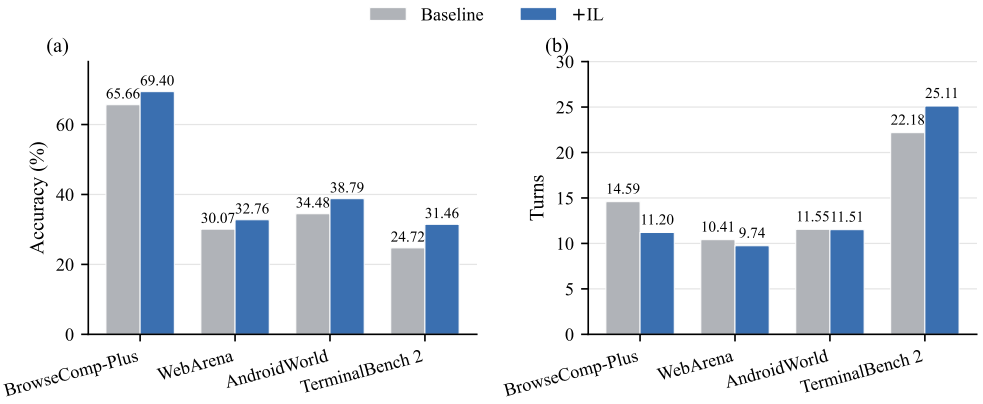


Figure 5: Inference lookahead with DeepSeek-V4-Pro. (a) Accuracy and (b) average turns over successful trajectories, comparing the baseline policy and the world-model-guided policy (+IL).

Table 2: Simulation SFT (+SSFT) and simulation RL (+SRL) on BFCL-v3 Team (2024) and tau2 Barres et al. (2025) for Qwen3.5-4B. The two stages are run separately; each row is compared against the Prompt baseline.

	BFCL-v3					tau2		
	Non-Live	Live	Multi-Turn	Hallu.	Relev.	Airline	Retail	Telecom
Prompt	36.65	31.46	6.75	42.32	62.50	40.00	33.33	0.00
+SSFT	40.92$_{+4.27}$	37.23$_{+5.77}$	28.00$_{+21.25}$	62.17$_{+19.85}$	81.25$_{+18.75}$	48.00$_{+8.00}$	42.98$_{+9.65}$	16.67$_{+16.67}$
+SRL	46.23$_{+9.58}$	41.15$_{+9.69}$	19.88$_{+13.13}$	63.32$_{+21.00}$	81.25$_{+18.75}$	38.00$_{-2.00}$	65.79$_{+32.46}$	16.67$_{+16.67}$

We also evaluate simulation SFT, simulation RL, and inference lookahead with WIA-WM. Simulation experiments use Qwen3.5-4B, and inference lookahead uses DeepSeek-V4-Pro as the policy. Table 2 reports simulation SFT and RL on BFCL-v3 and tau2, and Figure 5 reports inference lookahead. Simulation SFT (+SSFT) improves BFCL Multi-Turn, Hallucination, and Relevance by +21.25, +19.85, and +18.75 over Prompt, and improves tau2 Airline, Retail, and Telecom by +8.00, +9.65, and +16.67, lifting the BFCL-v3 overall by +13.98 and the tau2 overall by +11.44. The BFCL-v3 and tau2 +SSFT columns use different SFT variants (Appendix A). Simulation RL (+SRL) improves BFCL Non-Live, Live, Multi-Turn, Hallucination, and Relevance by +9.58, +9.69, +13.13, +21.00, and +18.75, and improves tau2 Retail and Telecom by +32.46 and +16.67, lifting the BFCL-v3 overall by +14.43 and the tau2 overall by +15.71. Inference lookahead improves accuracy on four benchmarks, with the largest gains on TerminalBench 2 Merrill et al. (2026) (+6.74) and AndroidWorld Rawles et al. (2025)ed (+4.31), and reduces average turns on BrowseComp-Plus Chen et al. (2025b) by 3.39 and on WebArena Zhou et al. (2024) by 0.67. All benchmarks are OOD relative to both training sets. Implementation details and additional curves are provided in Appendix A and Appendix E, respectively.

5.4 WIA-WAM Results

Table 3 reports the results on WebShop Yao et al. (2022), Search-R1 Jin et al. (2025), and ScienceWorld Wang et al. (2022). BC means behavioral cloning and ALEE means agent learning via early experience (Zhang et al., 2025). Among 4B variants, **WIA-WAM (WIA-CPT-4B + DPI + CSPO)** achieves the best performance across all three benchmarks. It improves Search-R1 accuracy by +1.56 percentage points, WebShop success rate by +3.13 percentage points, and ScienceWorld

Table 3: Performance comparison on three benchmarks. Higher is better except Turns (↓).

Backbone	Method	WebShop		Search-R1		ScienceWorld
		Score	Success Rate	Accuracy	Turns ↓	Success Rate
Qwen3.5-4B	BC + GRPO	91.50	75.00	50.00	5.50	42.20
	ALEE + GRPO	92.37	77.34	50.00	7.27	41.41
WIA-CPT-4B	BC + GRPO	91.12	75.00	50.39	5.14	51.58
	DPI + CSPO (**WIA-WAM**)	**93.21**	**78.13**	**51.95**	**4.98**	**53.73**
Qwen3.5-9B	BC + GRPO	95.55	83.59	51.17	5.35	71.88
	ALEE + GRPO	92.36	82.81	49.61	6.63	44.53
WIA-CPT-9B	BC + GRPO	91.37	74.20	51.95	**4.62**	70.31
	DPI + CSPO (**WIA-WAM**)	**96.65**	**84.38**	**52.34**	5.57	**75.00**

success rate by $+2.15$ percentage points over the **WIA-CPT-4B + BC + GRPO** ablation. Compared with the **Qwen3.5-4B + BC + GRPO** baseline, the success rate on ScienceWorld increases by $+11.53$ percentage points. The 9B comparison shows a consistent trend: WIA-WAM attains the highest Search-R1 accuracy (52.34%), WebShop score and success rate (96.65/84.38%), and ScienceWorld success rate (75.00%). Full training configurations are in Appendix A. Appendix F provides three side-by-side Search-R1 trajectories illustrating how WIA-WAM anticipates the next search interaction and grounds its answer more directly than the policy-only baselines.

5.5 Ablation Study

Table 4 reports the OOD ablation for the 4B, 9B, and 35B-A3B families under the DeepSeek-V4-Pro judge. `Avg` improves steadily across the three training stages: WIA-CPT improves over the base model by **3.50**, **1.82**, and **3.47**, WIA-SFT further improves `Avg` by **4.60**, **6.61**, and **9.56**, and WIA-WM further improves `Avg` by **2.50**, **7.26**, and **7.76** for the 4B, 9B, and 35B-A3B families, respectively, yielding total gains of **10.60**, **15.69**, and **20.79** over the base models. The gains are domain-dependent: Search and SWE benefit most, whereas Android improves less consistently, with a decrease at 4B. The remaining 27B OOD ablation and the ID results are in Appendix B.

Table 4: Ablation on OOD under the DeepSeek-V4-Pro judge for the 4B, 9B, and 35B-A3B models.

Stage	Android	MCP	OS	Search	SWE	Terminal	Web	Avg
Qwen3.5-4B-Base	35.88	33.29	47.07	26.43	33.14	36.00	40.21	35.74
WIA-CPT-4B	39.07	35.27	59.75	28.57	34.84	37.52	42.69	39.24
WIA-SFT-4B	37.82	39.14	44.81	34.40	49.57	51.58	46.88	43.84
WIA-WM-4B	**27.19**	**44.10**	**48.40**	**36.84**	**60.06**	**55.87**	**49.27**	**46.34**
Qwen3.5-9B-Base	37.12	36.88	53.45	27.27	39.56	39.08	40.80	38.80
WIA-CPT-9B	42.94	34.23	50.69	34.17	35.91	38.54	47.38	40.62
WIA-SFT-9B	39.45	43.69	57.51	37.82	50.59	51.96	49.45	47.23
WIA-WM-9B	**46.82**	**52.12**	**62.70**	**46.06**	**60.77**	**59.38**	**53.43**	**54.49**
Qwen3.5-35B-A3B-Base	35.08	31.83	43.62	27.35	33.70	36.21	37.41	34.93
WIA-CPT-35B-A3B	30.04	39.82	56.37	24.48	32.71	41.39	49.20	38.40
WIA-SFT-35B-A3B	43.69	40.08	52.21	40.13	50.16	54.33	51.59	47.96
WIA-WM-35B-A3B	**48.76**	**43.72**	**63.86**	**51.33**	**59.19**	**62.52**	**54.95**	**55.72**

6 Conclusion

We presented the **World Intelligence Agent (WIA)**, a unified framework that bridges environment prediction and action execution in digital agents. To establish a coherent foundation for end-to-end development, we introduced **WIA-Corpus** and **WIA-Bench**, standardizing both multi-domain training and generalization evaluation. Building on this, we developed **WIA-WM**, a multi-scale foundation world model that serves as a core cognitive engine for systematic state-transition modeling. Rather than treating the world model as an isolated artifact, we also introduced **WIA-WAM**, which couples predictive modeling and action generation via DPI and CSPO. By enabling the policy to internalize environment dynamics, WIA equips agents to act with a predictive understanding of the consequences of their actions, moving beyond reactive behavior. Our framework offers a principled and scalable direction for building more robust, forward-thinking digital agents.

AI USE STATEMENT

In this work, we used generative AI tools to polish our manuscript to improve the readability. We have not used generative AI tools for help develop theoretical models or conceptual frameworks, design or provide feedback on research methodology or experiments, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, interpret results. Formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, implement methods are not applicable to this work. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The main paper describes the training pipeline and evaluation protocol. The Appendix provides more details. The supplementary material provides the source code for WIA, covering WIA-WM and WIA-WAM.

REFERENCES

Wasi Uddin Ahmad, Nikolai Ludwig, Somshubra Majumdar, and Boris Ginsburg. Open-swe-traces: Advancing dual-mode multilingual distillation for software engineering agents. *arXiv preprint arXiv:2606.16038*, 2026.

Anthropic. Claude Opus 4.8 model. `https://www.anthropic.com/`, 2026a. Model documentation and release page.

Anthropic. Claude Opus 5 model. `https://www.anthropic.com/`, 2026b. Model documentation and release page.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-Bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.

Yannick Bleyenheuft, Philippe Lefevre, and Jean-Louis Thonnard. Predictive mechanisms control grip force after impact in self-triggered perturbations. *Journal of Motor Behavior*, 41(5):411–417, 2009.

Hyungjoo Chae, Namyoung Kim, Kai Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim, Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. Web agents with world models: Learning and leveraging environment dynamics in web navigation. In *International Conference on Learning Representations*, volume 2025, pp. 63707–63738, 2025.

Shiqi Chen, Tongyao Zhu, Zian Wang, Jinghan Zhang, Kangrui Wang, Siyang Gao, Teng Xiao, Yee Whye Teh, Junxian He, and Manling Li. Internalizing world models via self-play finetuning for agentic rl. *arXiv preprint arXiv:2510.15047*, 2025a.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, et al. Browsecomp-plus: A more fair and transparent evaluation benchmark of deep-research agent. *arXiv preprint arXiv:2508.06600*, 2025b.

Andy Clark et al. Whatever next? predictive brains, situated agents, and the future of cognitive science. *Behavioral and brain sciences*, 36(3):181–204, 2013.

Hang Ding, Peidong Liu, Junqiao Wang, Ziwei Ji, Meng Cao, Rongzhao Zhang, Lynn Ai, Eric Yang, Tianyu Shi, and Lei Yu. Dynaweb: Model-based reinforcement learning of web agents. *arXiv preprint arXiv:2601.22149*, 2026.

Tianqing Fang, Hongming Zhang, Zhisong Zhang, Kaixin Ma, Wenhao Yu, Haitao Mi, and Dong Yu. Webevolver: Enhancing web agent self-improvement with co-evolving world model. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 8970–8986, 2025.

Karl Friston. The free-energy principle: a unified brain theory? *Nature reviews neuroscience*, 11 (2):127–138, 2010.

Longxi Gao, Li Zhang, Shihe Wang, Shangguang Wang, Yuanchun Li, and Mengwei Xu. Mobileviews: A large-scale mobile gui dataset. *arXiv e-prints*, pp. arXiv–2409, 2024.

Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, et al. Is your llm secretly a world model of the internet? model-based planning for web agents. *arXiv preprint arXiv:2411.06559*, 2024.

Ananya Gubbi Mohanbabu, Rosiana Natalie, Brandon Kim, Anhong Guo, and Amy Pavel. A11y-cua dataset: Characterizing the accessibility gap in computer use agents. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*, pp. 1–26, 2026.

Shangmin Guo, Omar Darwiche Domingues, Raphaël Avalos, Aaron Courville, and Florian Strub. World modelling improves language model agents. *arXiv preprint arXiv:2506.02918*, 2025.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3):440, 2018.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Nick Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, volume 2024, pp. 47376–47405, 2024.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 8154–8173, 2023.

Hamish Ivison, Junjie Oscar Yin, Rulin Shao, Teng Xiao, Nathan Lambert, and Hannaneh Hajishirzi. Tmax: A simple recipe for terminal agents. *arXiv preprint arXiv:2606.23321*, 2026.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. arXiv preprint arXiv:2503.09516, 2025. URL `https://arxiv.org/abs/2503.09516`.

Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on ui control agents. *Advances in Neural Information Processing Systems*, 37:92130–92154, 2024.

Zhuofeng Li, Dongfu Jiang, Xueguang Ma, Haoxiang Zhang, Ping Nie, Yuyu Zhang, Kai Zou, Jianwen Xie, Yu Zhang, and Wenhu Chen. Openresearcher: A fully open pipeline for long-horizon deep research trajectory synthesis. *arXiv preprint arXiv:2603.20278*, 2026.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024.

Ning Lu, Baijiong Lin, Shengcai Liu, Jiahao Wu, Haoze Lv, Yanbin Wei, Lingting Zhu, Shengju Qian, Xin Wang, Ying-Cong Chen, et al. Policy and world modeling co-training for language agents. *arXiv preprint arXiv:2606.02388*, 2026a.

Shuo Lu, Yinuo Xu, Kecheng Yu, Siru Jiang, Yongcan Yu, Yubin Wang, Haitao Yang, Yuxiang Zhang, Bin Wang, Ran He, et al. Worldcoder-bench: Benchmarking physically grounded 3d world synthesis. *arXiv preprint arXiv:2606.01869*, 2026b.

Xing Han Lù and Siva Reddy. Structured distillation of web agent capabilities enables generalization. *arXiv preprint arXiv:2604.07776*, 2026.

Xing Han Lù, Zdeněk Kasner, and Siva Reddy. Weblinx: Real-world website navigation with multi-turn dialogue. *arXiv preprint arXiv:2402.05930*, 2024.

Nikolai Ludwig, Wasi Uddin Ahmad, Somshubra Majumdar, and Boris Ginsburg. From swe-zero to swe-hero: Execution-free to execution-based fine-tuning for software engineering agents. *arXiv preprint arXiv:2604.01496*, 2026.

Dezhao Luo, Bohan Tang, Kang Li, Georgios Papoudakis, Jifei Song, Shaogang Gong, Jianye Hao, Jun Wang, and Kun Shao. Vimo: A generative visual gui world model for app agents. In *International Conference on Learning Representations*, volume 2026, pp. 117437–117463, 2026.

Ziyang Luo, Zhiqi Shen, Wenzhuo Yang, Zirui Zhao, Prathyusha Jwalapuram, Amrita Saha, Doyen Sahoo, Silvio Savarese, Caiming Xiong, and Junnan Li. Mcp-universe: Benchmarking large language models with real-world model context protocol servers. *arXiv preprint arXiv:2508.14704*, 2025.

Mike Merrill, Alexander Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Shin, Thomas Walshe, E Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations*, volume 2026, pp. 40903–40986, 2026.

MiMo. MiMo-v2.5-pro. `https://mimo.xiaomi.com/`, 2026. Model documentation and release page.

OpenAI. GPT-5.6 model. `https://openai.com/`, 2026. Model documentation and release page.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with swe-gym. *arXiv preprint arXiv:2412.21139*, 2024.

Qwen Team. Qwen3.8-Max model. `https://qwen.ai/`, 2026. Model documentation and release page.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: international conference for high performance computing, networking, storage and analysis*, pp. 1–16. IEEE, 2020.

Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. Androidworld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Learning Representations*, volume 2025, pp. 406–441, 2025.

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, 2024.

Yueqi Song, Ketan Ramaneti, Zaid Sheikh, Ziru Chen, Boyu Gou, Tianbao Xie, Yiheng Xu, Danyang Zhang, Apurva Gandhi, Fan Yang, et al. Agent data protocol: Unifying datasets for diverse, effective fine-tuning of llm agents. In *International Conference on Learning Representations*, volume 2026, pp. 143153–143173, 2026.

Bohan Tang, Dezhao Luo, Jianheng Liu, Jingxuan Chen, Shaogang Gong, Jianye Hao, Jun Wang, and Kun Shao. Beyond syntax: Action semantics learning for app agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9444–9454, 2026.

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

The Gorilla Team. Berkeley function calling leaderboard v3: Multi-turn and multi-step function calling. `https://gorilla.cs.berkeley.edu/blogs/13_bfcl_v3_multi_turn.html`, 2024.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. Openhands: An open platform for ai software developers as generalist agents. In *International Conference on Learning Representations*, volume 2025, pp. 65882–65919, 2025.

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. *arXiv preprint arXiv:2602.10090*, 2026a.

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, 2026b. URL https://arxiv.org/abs/2602.10090.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Zikai Xiao, Jianhong Tu, Chuhang Zou, Yuxin Zuo, Zhi Li, Peng Wang, Bowen Yu, Fei Huang, Junyang Lin, and Zuozhu Liu. Webworld: A large-scale world model for web agent training. *arXiv preprint arXiv:2602.14721*, 2026.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

John Yang, Kilian Lieret, Carlos Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. Swe-smith: Scaling data for software engineering agents. *Advances in Neural Information Processing Systems*, 38, 2026.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

Xiao Yu, Baolin Peng, Ruize Xu, Michel Galley, Hao Cheng, Suman Nath, Jianfeng Gao, and Zhou Yu. Dyna-think: Synergizing reasoning, acting, and world model simulation in ai agents. *arXiv preprint arXiv:2506.00320*, 2025.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, et al. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*, 2025.

Xuan Zhang, Wenxuan Zhang, See-Kiong Ng, and Yang Deng. Self-evolving world models for llm agent planning. *arXiv preprint arXiv:2606.30639*, 2026.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization, 2025. URL https://arxiv.org/abs/2507.18071.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583, 2024.

Zhipu AI. GLM-5.3 model. https://www.zhipuai.cn/, 2026. Model documentation and release page.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024.

Yuxin Zuo, Zikai Xiao, Li Sheng, Fei Huang, Jianhong Tu, Yuxuan Liu, Tianyi Tang, Xiaomeng Hu, Yang Su, Qingfeng Lan, et al. Qwen-agentworld: Language world models for general agents. *arXiv preprint arXiv:2606.24597*, 2026.

A CONFIGURATION DETAILS

Training and evaluation configuration. Datasets and objectives. We use the CPT, SFT, and RL datasets from **WIA-Corpus** (Section 3). CPT uses both the full-sequence next-token objective and the observation-only objective, while SFT uses only the observation-only objective. All stages use BF16 mixed precision and full-parameter training.

Backbones and initialization. We build **WIA-CPT**, a shared continual-pretrained base, by training the Qwen3.5-4B, Qwen3.5-9B, Qwen3.5-27B dense, and Qwen3.5-35B-A3B MoE backbones on WIA-Corpus. Both **WIA-WM** and **WIA-WAM** are initialized from WIA-CPT; WIA-WM is then trained with world-model SFT followed by world-model RL, while WIA-WAM is trained with DPI SFT followed by CSPO RL.

CPT configuration. CPT uses a common configuration across scales: sequence length 32,768, global batch size 1,024, and AdamW with $\beta_1 = 0.9$ and $\beta_2 = 0.95$. The learning rate is 2×10^{-5}, following a cosine schedule with 3% warmup, and weight decay is 0.1. Models train for two epochs.

WIA-WM SFT configuration. WIA-WM SFT adopts the same hyperparameters across scales: AdamW, global batch size 128, sequence length 81,920, and weight decay 0.1. The learning rate is 10^{-5}, following a cosine schedule with 3% warmup. Models train for two epochs.

WIA-WM RL configuration. WIA-WM RL training uses one epoch at each scale with AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.98$), weight decay 0.1, and a constant learning rate of 10^{-6}. We use GSPO with clipping $\epsilon_{\text{clip}} = 3.5 \times 10^{-4}$ and set the KL and entropy coefficients to zero. Rollouts use temperature 1.0 and top-$p = 1.0$, with sequence length 40,960, prompt limit 32,768, response limit 8,192, a global batch size of 2,048, and eight samples per prompt. Actor training and SGLang rollout (Zheng et al., 2024) are placed on separate GPU sets. Evaluation uses a prompt budget of 65,536, a context length of 81,920, and a response limit of 16,384; models are evaluated periodically during training. The detailed parallelism settings for the three stages are provided below.

Reward and evaluation models. RL training for **WIA-WM** uses `DeepSeek-V4-Flash` as the world-model judge. For **WIA-WAM**, CSPO computes the world reward deterministically from the counterfactual continuation outcome, and does not use an LLM judge or an external reward model during training. The final **WIA-Bench** results in Section 5.2 are scored by `DeepSeek-V4-Pro`; Appendix B additionally reports robustness under the `DeepSeek-V4-Flash` judge.

Training parallelism configurations. For CPT, the 4B, 9B, 27 B, and 35B-A3B runs initialize from the pre-trained backbones `Qwen3.5-4B-Base`, `Qwen3.5-9B-Base`, `Qwen3.5-27B`, and `Qwen3.5-35B-A3B-Base`, respectively. All scales use a global batch size of $1{,}024$ with a micro-batch size of 1. The 4B run uses 128 GPUs with ms-swift and DeepSpeed ZeRO-1, SP$=4$, DP$=32$, and gradient accumulation 32; the 9B run uses 64 GPUs with Megatron-SWIFT, TP$=4$/PP$=1$/CP$=1$, DP$=16$, and gradient accumulation 64; the 27B run uses 128 GPUs with Megatron-SWIFT, TP$=8$/PP$=1$/CP$=1$, DP$=16$, and gradient accumulation 64; and the 35B-A3B MoE run uses 128 GPUs with Megatron-SWIFT, TP$=2$/PP$=8$/EP$=8$/ETP$=1$, DP$=8$, and gradient accumulation 128. For SFT, all scales use 8 GPUs with a global batch size of 128 and a micro-batch size of 1. All runs use ms-swift with DeepSpeed ZeRO-3. For RL, all scales use Miles with a Megatron-LM training backend and a global batch size of $2{,}048$ (256 prompts with $n = 8$ rollout samples per prompt). The 4B run uses 32 actor GPUs (TP$=2$, PP$=1$, CP$=4$) and 32 rollout GPUs; the 9B run uses 64 actor GPUs (TP$=2$, PP$=2$, CP$=4$) and 64 rollout GPUs; the 27B dense run uses 64 actor GPUs (TP$=4$, PP$=4$, CP$=4$) and 64 rollout GPUs; and the 35B-A3B MoE run uses 64 actor GPUs (TP$=2$, PP$=4$, CP$=4$, EP$=2$) and 64 rollout GPUs. Rollout uses one SGLang engine per GPU for the 4B model and one engine per two GPUs for the larger models. The DeepSeek-V4-Flash world-model judger is served on 16 GPUs.

World-model evaluation. All WIA-WM checkpoints are evaluated on WIA-Bench ($1{,}808$ ID and $1{,}739$ OOD examples across seven domains) with a $65{,}536$-token prompt budget, an $81{,}920$-token context length, and a $16{,}384$-token response limit. Rollout decoding uses temperature 1.0 and top-p 1.0; prompts are left-truncated to preserve the latest history and the final `###[Observation]` turn, and generation stops at `<|im_end|>`. Each prediction is scored by an LLM judge over six dimensions—format, factuality, consistency, realism, quality, and efficiency—each scored in $[1, 5]$; the weighted mean with weights $(1, 2, 2, 2, 2, 1)$ is converted to a 20–100 scale by mul-

tiplying the five-point score by 20. Main results use `DeepSeek-V4-Pro` as the judge, with `DeepSeek-V4-Flash` as a robustness check, and we report the sample-weighted average (`Avg`) and the seven-domain macro average (`MAvg`). Let N be the number of samples in a split and N_j the number of samples in domain j (with $\sum_{j=1}^{7} N_j = N$), and let s_i be the percent-converted score of sample i and $\bar{s}_j = \frac{1}{N_j} \sum_{i \in \text{domain}_j} s_i$ the domain-j mean. Then $\texttt{Avg} = \frac{1}{N} \sum_{i=1}^{N} s_i = \sum_{j=1}^{7} \frac{N_j}{N} \bar{s}_j$, and $\texttt{MAvg} = \frac{1}{7} \sum_{j=1}^{7} \bar{s}_j$.

World-model ablation. Ablations reuse the identical rollout and judge protocol. For all four families—4B, 9B, 27B, and 35B-A3B—we compare the base backbone, the WIA-CPT stage, the world-model SFT stage, and the full WIA-WM model on both the ID and OOD suites, using `DeepSeek-V4-Pro` for the main numbers and `DeepSeek-V4-Flash` for robustness. This isolates the contributions of continual pre-training, SFT alignment, and world-model RL.

Simulation SFT. Simulation SFT uses WIA-WM as a synthetic environment: a teacher policy proposes actions, WIA-WM predicts the resulting observations, and the paired action-observation trajectories are used to fine-tune a student policy without further real-environment interaction. The training prompts are the `train_tools_lt30` subset of the Agent World Model benchmark (Wang et al., 2026a;b), which contains 2,310 MCP tasks whose task-local tool domain has at most 30 tools, drawn from the benchmark's full 9,500-task training split (whose tool domains reach up to 87 tools). For BFCL-v3 and tau2, trajectories are generated with the `deepseek-v4-pro` teacher (fallback `qwen3.8-max`) running the agent loop for up to 50 turns; the world model `WIA-WM-35B-A3B` is deployed on 32 GPUs and served through four SGLang endpoints with temperature 0.6, top-p 0.95, and a 4,096-token response limit, and each of the 2,310 tasks is rolled out for up to five attempts, yielding 11,288 completed trajectories. Each trajectory is scored by the `deepseek-v4-pro` teacher judge on the 0–5 scale; only trajectories with a judge score of at least 3 are kept and converted to the Qwen3-Coder inline-XML tool format. The converted sequences are tokenized with the `Qwen3.5-4B` chat template, and the longest 1% is dropped at the 99th-percentile length of 12,811 tokens, leaving 9,922 trajectories that are used to fine-tune the `Qwen3.5-4B` student from the base backbone. The student is trained full-parameter with ms-swift on 8 GPUs using DeepSpeed ZeRO-3, sequence parallelism 2, data parallelism 4, micro-batch size 1, gradient accumulation 32 (global batch 128), sequence length 16,384, two epochs, learning rate 10^{-5} with a cosine schedule and 3% warmup, weight decay 0.1, and gradient clipping 1.0. The BFCL-v3 and tau2 evaluations use a maximum of 20 interaction turns; per-subset results are reported as the +SSFT rows of Table 2. The BFCL-v3 and tau2 +SSFT columns in Table 2 use two different SFT variants. The BFCL-v3 columns use the 9,922 five-round `train_tools_lt30` trajectories described above, trained for two epochs. The tau2 columns use a second variant trained on the full AWM training set: in addition to those 9,922 trajectories, it adds 21,345 complement trajectories generated from the remaining tasks (whose task-local tool domain exceeds 30 tools) with a compact natural-language/tool-reference world-model prompt for up to three attempts, for 32,633 trajectories in total, trained for one epoch. Both variants share the same `deepseek-v4-pro` teacher, `WIA-WM-35B-A3B` world model, Qwen3-Coder inline-XML formatting, and optimization settings (global batch 128, sequence length 16,384, learning rate 10^{-5}, weight decay 0.1), differing only in task coverage, rollout rounds, and training epochs.

Simulation RL. Simulation RL trains the policy against WIA-WM instead of the real environment: the policy issues candidate actions, WIA-WM returns predicted observations, and the policy is optimized with the task reward. Training prompts use the same task set as Simulation SFT. For BFCL-v3 and tau2, the `Qwen3.5-4B` policy is trained with Miles/GSPO across 8 nodes—32 actor GPUs and 32 rollout GPUs—with tensor parallelism 2, pipeline parallelism 1, and context parallelism 4. Rollout uses temperature 1.0, a 16,384-token prompt and an 8,192-token response limit, 64 prompts per batch with 8 samples per prompt (global batch 512), and 20 turns per episode. Training runs three epochs with Adam ($\beta_1 = 0.9$, $\beta_2 = 0.98$, weight decay 0.1), a constant learning rate 10^{-6}, and GSPO clipping $\epsilon = 3.5 \times 10^{-4}$ with zero KL and entropy coefficients. The world model is the same `WIA-WM-35B-A3B`, deployed on 32 GPUs and served at temperature 0.6 and top-p 0.95. Each rollout is rewarded by $0.8\, r_{\text{task}} + 0.2\, r_{\text{fmt}}$, where the task term r_{task} is the `DeepSeek-V4-Flash` judge score on the complete trajectory against the benchmark criteria, rated 0–5 (success probability 80%, execution efficiency 20%) and rescaled to $[0, 1]$ by dividing

by 5, and the format term r_{fmt} is a binary signal that is 1 only when the final answer contains no residual tool-call block and every turn uses at most one tool call. The BFCL-v3 and tau2 evaluations use a maximum of 20 interaction turns; per-subset gains are reported as the +SRL rows of Table 2 (largest gains tau2 Retail +32.46 and BFCL Hallucination +21.00).

Inference lookahead. Inference lookahead applies WIA-WM at test time without retraining the policy. At each agent step the policy samples $k = 3$ candidate next actions, which are de-duplicated. For each distinct candidate, `WIA-WM-35B-A3B` (deployed on 32 GPUs, temperature 0.6) predicts the corresponding next observation conditioned on the goal, a short window of the most recent real action–observation pairs, and that candidate action. A selector (the same policy at temperature 0.0) then reads the goal, the real history, and each candidate together with its predicted observation, and chooses the single most promising action; only that action is executed in the real environment, and its observation is appended to the history. With `DeepSeek-V4-Pro` as the policy, this improves accuracy on TerminalBench 2 (+6.74, maximum 50 interaction turns), AndroidWorld (+4.31, maximum 20 turns), BrowseComp-Plus (+3.73, maximum 30 turns), and WebArena (+2.69, maximum 30 turns), as summarized in Figure 5.

WIA-WAM training configuration. For WebShop, before GRPO optimization we construct a dataset of 30,000 single-turn dialogue instances from expert trajectories, comprising 15,000 agent-view and 15,000 world-view samples. Each policy model is then initialized from its corresponding SFT variant. The SFT protocol enforces a maximum prompt context length of 8,192 tokens and a maximum response length of 1,024 tokens, and uses a learning rate of 1×10^{-5}. Each variant is trained for 2 epochs: the BC model uses behavior-cloning data, the ALEE model uses its official dataset, and the DPI model uses a 1:1 mixture of agent-view and world-view data, with strict trajectory-level isolation. Policy fine-tuning uses GRPO with 16 task prompts per iteration, $G = 16$ trajectories per prompt, and a global batch size of 64 over 150 iterations; trajectories are capped at 15 turns, prompt and action response lengths are bounded at 4,096 and 512 tokens, and the update uses Adam with learning rate 10^{-6}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay 0.01, symmetric clipping $\epsilon = 0.2$, and entropy coefficient 0.001.

For Search-R1, the main multi-turn DPI run uses 25,778 mixed views (12,889 agent-view and 12,889 world-view samples), whereas Expert SFT and WIA-CPT controls use only policy trajectories; ALEE uses one epoch of world SFT followed by one epoch of policy SFT. The reported Expert SFT model is trained with two policy-SFT epochs at sequence parallelism 4 followed by one epoch at sequence parallelism 2, while the WIA-CPT control is initialized from WIA-CPT-4B and trained with the standard two-epoch schedule. We also evaluate a matched transition-level format that renders targets as separate system/user/assistant samples. Both the multi-turn and transition-level SFT configurations perform 2 epochs of full-parameter BF16 training using AdamW (learning rate $= 10^{-5}$, weight decay 0.01, cosine schedule, 5% warmup), max context length 8,192, left truncation, no sequence packing, gradient clipping 1.0, per-device batch size 1, gradient accumulation 8, sequence parallelism 4, and DeepSpeed ZeRO-1 (Rajbhandari et al., 2020) (note that WIA-WAM SFT uses weight decay 0.01 rather than the 0.1 used in WIA-WM). For RL, WIA-WAM CSPO trains for 200 steps on a fixed 4,096-prompt pool. Each step samples 4 prompt groups, generating 8 agent, 4 world, and 4 continuation rollouts per group (64 samples/update). We use Adam ($\beta_1 = 0.9, \beta_2 = 0.98$, weight decay 0.1), constant learning rate 10^{-7}, rollout temperature 1.0, top-$p = 1.0$, max sequence length 16,384, prompt limit 8,192, context parallelism 4, gradient clipping 1.0, and GSPO clipping $\epsilon_{\text{clip}} = 3.5 \times 10^{-4}$ with zero KL/entropy coefficients. Response length limits are 512 tokens for policy/continuation and 1,024 for world predictions. The 4B RL setup uses four actor GPUs and four rollout GPUs, with the actor and rollout GPUs on separate GPU sets. Continuation rewards are computed from transcripts and retriever observations without learned reward models. Policy-only RL controls (Expert SFT and WIA-CPT) share the same RL hyperparameters, prompt pool, 200 steps, and 64-sample batch size, but sample 8 prompt groups (8 agent rollouts each, max 512 response tokens) with correctness-only rewards, omitting world and continuation branches.

For ScienceWorld, the DPI SFT preprocessing stage adopts the same learning rate of 1×10^{-5}. The BC and ALEE baselines are each fine-tuned for 2 epochs, with ALEE using its official dataset and process. DPI data are constructed from 4,000 multi-turn trajectories, blending 2,000 agent-view and 2,000 world-view samples at a 1:1 ratio with strict trajectory-level isolation. The DPI model is then trained for 2 epochs. Additionally, a data pre-filtering protocol is applied before RL training,

discarding task instances with gold-path trajectory lengths exceeding 60 steps to ensure exploration efficiency. During policy optimization, GRPO is configured to execute 50 rollout iterations with a rollout batch size of 16 task prompts, sampling $G = 8$ trajectories per prompt to construct a global batch size of 64. The multi-step interaction allows up to 55 environment steps per episode, and a sliding history context window retains the 8 most recent action–observation pairs. Prompt and single-step response lengths are constrained to 4,096 and 128 tokens, respectively. The advantage function strictly follows the standard GRPO formulation, using group-normalized final task scores ($G = 8$) without intermediate step-level advantage signals or KL regularization ($\beta_{\text{KL}} = 0$). Optimization is performed using Adam with a constant learning rate of 1×10^{-6}, $\beta_1 = 0.9, \beta_2 = 0.98$, and weight decay 0.1. An asymmetric clipping range of $[\epsilon_{\text{low}}, \epsilon_{\text{high}}] = [0.2, 0.28]$ is applied alongside an entropy coefficient of 0.0. Per-token loss computation is coupled with inter-GPU data balancing to mitigate computational imbalance arising from variable trajectory lengths.

B Supplementary Results

Table 5 reports the full ID results under the `DeepSeek-V4-Pro` judge. WIA-WM achieves the two highest sample-weighted averages, **66.41** for **WIA-WM-27B** and **65.04** for **WIA-WM-35B-A3B**, ahead of the strongest baseline Claude-Opus-4.8 (55.76); **WIA-WM-9B** reaches **61.99**. On ID, `Avg` rises with dense scale from 59.17 (4B) to 66.41 (27B), and the MoE **WIA-WM-35B-A3B** is close behind at 65.04.

Compared with the same-size Qwen3.5 backbones, WIA-WM improves ID `Avg` by +16.97 (4B, 42.20 to 59.17), +19.60 (9B, 42.39 to 61.99), +17.50 (27B, 48.91 to 66.41), and +19.57 (35B-A3B, 45.47 to 65.04). The gains are largest on Search and Terminal; SWE also improves by roughly +22 to +24 points, and MCP improves by +8.8 to +10.7. **WIA-WM-4B** already scores 82.59 on Search and 75.76 on Terminal, and these rise to 88.54 and 80.85 for **WIA-WM-27B**.

Table 5: In-domain world-model results under the DeepSeek-V4-Pro judge. Red and blue bold values indicate the best and second-best results in each column, respectively. `Avg` is the sample-weighted average across all samples, while `MAvg` is the macro average across the seven domains.

	Model	Android	MCP	OS	Search	SWE	Terminal	Web	Avg	MAvg
Proprietary	GPT-5.6-sol	**43.62**	61.02	33.08	50.66	53.11	59.96	62.38	52.04	51.98
	GPT-5.6-terra	41.57	61.98	33.08	47.32	52.81	59.25	60.94	51.05	50.99
	GPT-5.6-luna	41.29	60.03	32.43	45.40	51.29	55.28	58.33	49.11	49.15
	Claude-Opus-4.8	41.59	59.08	33.00	63.50	56.46	65.91	66.21	55.76	55.11
	Claude-Sonnet-5	40.08	60.28	32.27	58.46	38.72	60.40	63.96	50.75	50.59
Open-source	GLM-5.3	41.20	58.63	33.20	54.74	32.09	63.22	64.52	49.64	49.66
	MiMo-v2.5-pro	40.93	59.61	32.39	51.34	47.36	54.35	64.16	50.12	50.02
	Qwen3.8-27B	36.69	55.67	30.75	50.87	43.50	52.67	61.15	47.59	47.33
	Qwen3.8-max	41.18	58.32	32.39	61.00	51.06	65.09	65.82	54.06	53.55
	DeepSeek-V4-Flash	40.64	59.23	32.39	56.41	44.87	56.97	64.40	50.90	50.70
	DeepSeek-V4-Pro	41.54	59.06	32.31	55.72	45.65	55.51	64.51	50.76	50.61
Qwen3.5	Qwen3.5-4B-base	32.60	50.10	29.70	36.88	29.56	32.09	48.20	36.70	37.02
	Qwen3.5-4B	35.61	54.57	30.96	45.63	33.14	40.22	56.33	42.20	42.35
	Qwen3.5-9B-base	33.58	55.68	30.93	37.35	32.54	25.91	49.68	37.48	37.95
	Qwen3.5-9B	36.80	55.09	31.81	38.29	37.16	38.97	60.23	42.39	42.62
	Qwen3.5-35B-A3B-base	29.99	49.58	28.33	33.21	30.20	38.06	45.86	36.26	36.46
	Qwen3.5-35B-A3B	37.50	56.75	32.21	39.15	41.76	50.03	61.36	45.47	45.54
	Qwen3.5-27B	41.24	57.26	32.09	45.92	43.77	57.37	64.72	48.91	48.91
	Qwen-AgentWorld	39.03	56.99	32.49	63.68	45.60	61.33	64.54	52.51	51.95
	WIA-WM-4B	27.12	65.26	30.76	82.59	57.52	75.76	63.24	59.17	57.46
	WIA-WM-9B	41.61	63.89	32.79	84.31	60.87	73.96	68.69	61.99	60.87
	WIA-WM-27B	43.21	**67.74**	**35.28**	**88.54**	**66.24**	**80.85**	**73.84**	**66.41**	**65.10**
	WIA-WM-35B-A3B	**43.49**	**67.45**	**34.39**	**89.30**	**64.18**	**76.36**	**71.86**	**65.04**	**63.86**

Table 6 reports the same ID suite under the `DeepSeek-V4-Flash` judge. **WIA-WM-27B** and **WIA-WM-35B-A3B** again rank first and second, at **69.16** and **67.73** (`Avg`), and **WIA-WM-9B**

Table 6: In-domain world-model results under the DeepSeek-V4-Flash judge. Red and blue bold values indicate the best and second-best results in each column, respectively. `Avg` is the sample-weighted average across all samples, while `MAvg` is the macro average across the seven domains.

	Model	Android	MCP	OS	Search	SWE	Terminal	Web	Avg	MAvg
Proprietary	GPT-5.6-sol	46.66	64.93	34.62	52.60	52.54	61.19	67.64	54.23	54.31
	GPT-5.6-terra	44.69	65.93	34.44	50.44	51.14	59.81	66.06	53.11	53.22
	GPT-5.6-luna	44.24	63.53	33.93	47.41	49.64	56.37	63.40	51.03	51.22
	Claude-Opus-4.8	45.27	60.96	34.57	65.31	56.90	68.11	69.92	57.83	57.29
	Claude-Sonnet-5	45.27	62.98	34.30	59.93	41.02	62.23	68.01	53.37	53.39
Open-source	GLM-5.3	42.51	61.33	34.69	56.20	32.20	65.95	68.78	51.65	51.67
	MiMo-v2.5-pro	46.20	62.46	34.20	53.13	46.58	54.53	68.01	51.99	52.16
	Qwen3.8-27B	41.92	57.97	31.95	53.19	41.99	54.85	65.46	49.65	49.62
	Qwen3.8-max	45.78	61.33	34.03	62.33	49.99	66.81	70.13	56.06	55.77
	DeepSeek-V4-Flash	45.24	61.19	34.54	59.04	44.20	59.65	68.25	53.21	53.16
	DeepSeek-V4-Pro	46.19	62.01	34.18	56.91	45.65	58.17	68.23	52.99	53.05
Qwen3.5	Qwen3.5-4B	41.20	56.88	32.77	48.16	32.67	39.91	60.68	44.22	44.61
	Qwen3.5-9B	42.83	57.24	33.64	40.17	35.90	38.83	63.63	44.07	44.61
	Qwen3.5-35B-A3B	44.25	59.67	33.88	41.60	40.84	50.45	65.45	47.62	48.02
	Qwen3.5-27B	46.60	59.71	33.66	47.90	41.65	59.69	68.54	50.85	51.11
	Qwen-AgentWorld	45.96	60.82	34.16	65.55	45.48	63.94	68.91	55.19	54.97
	WIA-WM-4B	31.45	67.74	32.12	85.88	61.51	77.38	68.28	62.27	60.62
	WIA-WM-9B	46.41	66.70	34.85	86.31	63.84	74.78	71.40	64.43	63.47
	WIA-WM-27B	**49.13**	**70.00**	**38.29**	**90.62**	**68.12**	**82.60**	**77.54**	**69.16**	**68.04**
	WIA-WM-35B-A3B	**48.32**	**70.24**	**37.06**	**91.06**	**66.41**	**78.35**	**75.44**	**67.73**	**66.70**

reaches 64.43, above every baseline. The ordering is the same as under `DeepSeek-V4-Pro`, and the gap between **WIA-WM-27B** and Claude-Opus-4.8 is +11.33 points under Flash versus +10.65 under Pro. Flash scores are higher across the board: +3.10 (4B), +2.44 (9B), +2.75 (27B), and +2.69 (35B-A3B) for WIA-WM, and +2.07 for Claude-Opus-4.8.

Table 7 reports the OOD suite under the `DeepSeek-V4-Flash` judge. **WIA-WM-35B-A3B** and **WIA-WM-27B** remain the top two models at **60.09** and **59.50** (`Avg`), and WIA-WM improves with scale from 50.29 (4B) to 60.09 (35B-A3B); **WIA-WM-9B** reaches 58.46, ahead of the strongest baseline Claude-Opus-4.8 (56.89).

Per-domain, **WIA-WM-35B-A3B** leads Search by a wide margin (55.93 versus 48.24 for Qwen-AgentWorld and 40.38 for Claude-Opus-4.8) and Web (60.30 versus 59.87 for Claude-Opus-4.8), **WIA-WM-27B** leads SWE (62.16 versus 60.07 for Claude-Opus-4.8), and on Terminal GPT-5.6-sol remains first (68.30 versus 66.36 for WIA-WM-35B-A3B).

Tables 8 and 9 report the additional ablation on the OOD and ID suites under the `DeepSeek-V4-Pro` judge.

Table 8 reports the additional OOD ablation for the 27B model. The full **WIA-WM** model improves over SFT by +9.09, and the RL stage contributes most for the 27B model, where `Avg` moves from 46.36 (SFT) to 55.45.

Table 9 reports the ID ablation for all four model scales. The same trend is stronger on Search and Terminal: SFT lifts Search from 36.88 to 81.72 for the 4B model and from 45.92 to 83.30 for the 27B model, and the full **WIA-WM** model reaches 75.76 and 80.85 on Terminal. At 4B, RL further improves Search, Terminal, and SWE and keeps `Avg` at 59.17, comparable to the SFT-only model (59.47); at 9B and larger scales, RL adds +4.44 (9B), +4.45 (27B), and +5.51 (35B-A3B). WIA-CPT improves over the base model by +4.00 (4B, 36.70 to 40.70), +3.92 (9B, 37.48 to 41.40), and +2.23 (35B-A3B, 36.26 to 38.49), while the 27B CPT run is 5.02 points below its base model (48.91 to 43.89); its full contribution is realized together with SFT and world-model RL.

Figure 6 shows the WIA-WM sample-weighted averages on the ID and OOD suites across the four model scales under the `DeepSeek-V4-Pro` judge.

Table 7: Out-of-domain world-model results under the DeepSeek-V4-Flash judge. Red and blue bold values indicate the best and second-best results in each column, respectively. `Avg` is the sample-weighted average across all samples, while `MAvg` is the macro average across the seven domains.

	Model	Android	MCP	OS	Search	SWE	Terminal	Web	Avg	MAvg
Proprietary	GPT-5.6-sol	54.59	46.45	65.48	39.04	58.42	**68.30**	59.73	56.43	56.00
	GPT-5.6-terra	54.09	44.63	59.94	39.56	57.10	66.68	58.83	54.99	54.40
	GPT-5.6-luna	52.28	45.29	57.82	40.32	55.28	**66.73**	57.90	54.19	53.66
	Claude-Opus-4.8	53.20	**56.53**	68.93	40.38	60.07	62.13	59.87	56.89	57.30
	Claude-Sonnet-5	53.99	52.00	68.81	38.11	47.09	50.49	58.34	51.97	52.69
Open-source	GLM-5.3	49.02	52.33	60.96	31.62	31.83	55.13	56.76	47.11	48.23
	MiMo-v2.5-pro	51.50	46.35	66.29	40.37	49.30	57.63	56.04	52.46	52.50
	Qwen3.8-27B	45.07	41.58	58.16	35.71	46.97	55.92	50.85	47.88	47.75
	Qwen3.8-max	50.98	47.64	63.92	38.32	55.56	66.33	**60.25**	54.94	54.71
	DeepSeek-V4-Flash	50.83	46.61	65.16	40.44	49.56	59.51	55.35	52.49	52.49
	DeepSeek-V4-Pro	52.81	45.47	68.70	40.71	52.24	62.16	56.60	54.27	54.10
Qwen3.5	Qwen3.5-4B	45.01	40.53	61.85	32.35	37.75	41.83	49.69	43.67	44.14
	Qwen3.5-9B	47.28	43.34	63.58	32.88	43.97	45.44	52.38	46.56	46.98
	Qwen3.5-35B-A3B	53.06	43.44	62.94	33.52	45.34	52.06	53.61	48.93	49.14
	Qwen3.5-27B	52.78	43.93	**70.14**	34.55	47.04	58.97	56.28	51.85	51.96
	Qwen-AgentWorld	**54.80**	42.35	**70.67**	48.24	55.62	61.19	58.52	56.59	55.91
	WIA-WM-4B	30.97	46.99	61.36	39.82	58.73	59.19	53.96	50.29	50.15
	WIA-WM-9B	53.50	53.69	69.41	**49.37**	**61.22**	63.07	58.46	58.46	58.39
	WIA-WM-27B	**56.13**	**55.90**	68.69	49.28	**62.16**	65.96	58.38	**59.50**	**59.50**
	WIA-WM-35B-A3B	54.41	47.67	69.83	**55.93**	60.76	66.36	**60.30**	**60.09**	**59.32**

Table 8: Ablation on OOD under the DeepSeek-V4-Pro judge for the 27B model. Bold rows denote the full WIA-WM models.

Stage	Android	MCP	OS	Search	SWE	Terminal	Web	Avg
Qwen3.5-27B	47.84	42.56	64.29	33.75	48.10	55.09	52.78	49.15
WIA-CPT-27B	42.98	40.98	49.54	30.20	44.18	51.39	50.22	44.19
WIA-SFT-27B	44.18	43.26	43.26	33.27	53.34	54.69	50.48	46.36
WIA-WM-27B	**50.51**	**52.53**	**63.42**	**46.36**	**59.92**	**60.73**	**54.52**	**55.45**

C Scaling Analysis of WIA-WM

This section analyzes how WIA-WM scales, using the aggregate scaling curve (Figure 6), the per-domain RL validation curves (Figures 9 and 10), and the stage-wise ablations (Tables 4, 8, and 9).

ID keeps improving, while OOD saturates after 9B. As Figure 6 shows, most of the OOD gain comes from scaling 4B→9B (46.34 → 54.49, +8.15). Further scaling adds only +0.96 (27B) and +0.27 (35B-A3B). ID, in contrast, keeps improving up to 27B (66.41). The OOD plateau hides heterogeneous per-domain behavior (Table 1). SWE is already saturated at 4B (60.06 vs. 59.19 at 35B-A3B). Search (36.84 → 51.33) and Terminal (55.87 → 62.52) keep improving; both require reproducing long, knowledge-heavy outputs, where larger models hold more usable world knowledge. The MoE model activates only about 3B parameters per token, yet it outperforms the 4B dense model by +5.87 ID and +9.38 OOD, and it matches the 27B dense model on OOD. This suggests that total capacity matters more than activated compute for world modeling.

The RL stage benefits most from scale. SFT accounts for most of the ID gain at every scale (+16 to +21 points; Table 9). The gain from RL, however, grows with scale. At 4B, RL changes Avg by −0.30 on ID and +2.50 on OOD. From 9B upward, it adds +4.4 to +5.5 on ID and +7.3 to +9.1 on OOD (Tables 4 and 9). The validation curves show the same pattern (Figures 9, 10, and 11). For 9B and larger models, ID scores plateau within the first few evaluations, while OOD scores keep rising throughout training, most visibly on Search. The 4B run, in contrast, oscillates without a clear

Table 9: Ablation on ID under the DeepSeek-V4-Pro judge for the 4B, 9B, 27B, and 35B-A3B models. Bold rows denote the full WIA-WM models.

Stage	Android	MCP	OS	Search	SWE	Terminal	Web	Avg
Qwen3.5-4B-Base	32.60	50.10	29.70	36.88	29.56	32.09	48.20	36.70
WIA-CPT-4B	37.66	55.10	30.30	39.15	36.07	41.10	48.66	40.70
WIA-SFT-4B	38.34	61.70	33.46	81.72	53.83	68.84	70.33	59.47
WIA-WM-4B	**27.12**	**65.26**	**30.76**	**82.59**	**57.52**	**75.76**	**63.24**	**59.17**
Qwen3.5-9B-Base	33.58	55.68	30.93	37.35	32.54	25.91	49.68	37.48
WIA-CPT-9B	38.47	58.51	30.19	46.45	26.91	30.91	62.29	41.40
WIA-SFT-9B	38.87	60.28	31.37	77.19	52.09	70.94	65.40	57.55
WIA-WM-9B	**41.61**	**63.89**	**32.79**	**84.31**	**60.87**	**73.96**	**68.69**	**61.99**
Qwen3.5-27B	41.24	57.26	32.09	45.92	43.77	57.37	64.72	48.91
WIA-CPT-27B	38.09	59.84	29.03	36.27	43.14	38.37	65.01	43.89
WIA-SFT-27B	41.44	62.81	34.35	83.30	58.81	71.40	73.45	61.96
WIA-WM-27B	**43.21**	**67.74**	**35.28**	**88.54**	**66.24**	**80.85**	**73.84**	**66.41**
Qwen3.5-35B-A3B-Base	29.99	49.58	28.33	33.21	30.20	38.06	45.86	36.26
WIA-CPT-35B-A3B	28.45	56.07	30.24	27.91	32.56	32.24	62.99	38.49
WIA-SFT-35B-A3B	40.03	63.26	31.95	78.52	56.42	68.47	71.15	59.53
WIA-WM-35B-A3B	**43.49**	**67.45**	**34.39**	**89.30**	**64.18**	**76.36**	**71.86**	**65.04**

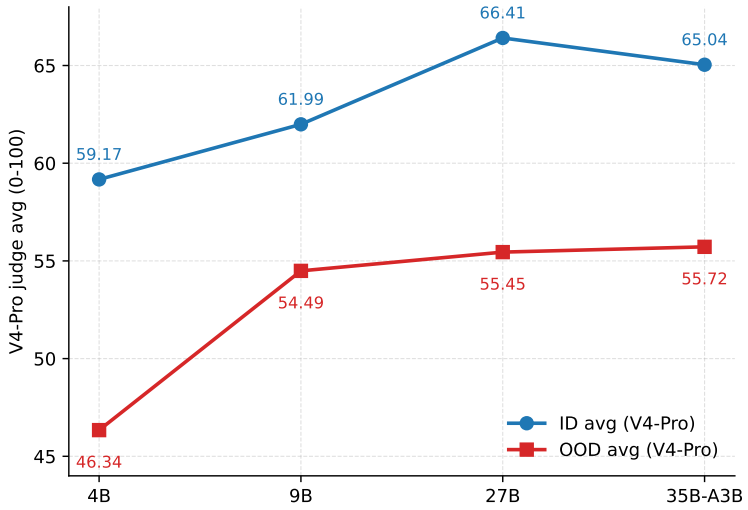


Figure 6: Scaling of the WIA-WM sample-weighted average on the ID and OOD suites under the DeepSeek-V4-Pro judge.

upward trend. Judge-based RL therefore mainly improves generalization, and only once the model is large enough.

The 4B regression on UI-state domains is caused by RL. WIA-WM-4B scores only 27.12 (ID) and 27.19 (OOD) on Android, below its own backbone. This is not a failure to learn Android. Before RL, the 4B and 9B models are nearly identical on Android (38.34 vs. 38.87 ID; 37.82 vs. 39.45 OOD). RL then lowers the 4B model by about 11 points and raises the 9B model. Across all scales, RL yields a negative domain-level change only at 4B, on Android (-11.22 ID, -10.63 OOD), Web (-7.09 ID), and OS (-2.70 ID). All 42 domain-level RL changes at 9B, 27B, and 35B-A3B are positive (Tables 4, 8, and 9). The regressing domains are exactly those whose observations are structured UI trees; domains with text or tool-output observations still improve at 4B. We attribute this to capacity competition within a single shared policy. The RL prompt pool is dominated by

Search (25.3%) and SWE (18.6%), and the judge rubric tolerates coarser-grained predictions. A small model can therefore trade node-level accuracy on UI trees for gains elsewhere.

D Dataset Construction and Distribution Details

This appendix reports the sources, processing pipelines, and filtering rules for all datasets used in continual pre-training (CPT), SFT, RL, and the out-of-domain (OOD) benchmark.

Shared interaction format and supervision targets. All records are rewritten as alternating `user` and `assistant` turns. Each turn is either a task goal, an action, or an environment observation, and every turn carries an explicit `loss` mask: system prompts, goals, and history/context turns use `loss=false`, while supervised targets use `loss=true`. Four target forms are used across the three training stages.

- **Continual pre-training, agent view.** A trajectory is arranged as $\text{goal} \rightarrow \text{action}_1 \rightarrow \text{observation}_1 \rightarrow \cdots \rightarrow \text{action}_n \rightarrow \text{observation}_n \rightarrow \text{answer}$. Actions, observations, and answer are both supervised, so the trajectory-level objective assigns a loss to each target.
- **Continual pre-training, world-model view.** A trajectory is arranged as $\text{observation}_0 \rightarrow \text{action}_1 \rightarrow \text{observation}_1 \rightarrow \cdots \rightarrow \text{action}_n \rightarrow \text{observation}_n$. Only the observation targets participate in the loss; actions remain in context with `loss=false`.
- **SFT.** High-quality multi-turn trajectories are arranged as $\text{goal} \rightarrow \text{action}_1 \rightarrow \text{observation}_1 \rightarrow \cdots \rightarrow \text{action}_n \rightarrow \text{observation}_n$. Only observations are supervised. Each action is rewritten into natural-language style while retaining its underlying tool call or command, so the action turns remain interpretable context rather than regression targets.
- **RL.** Each prompt is turn-level: $\text{history} + \text{current action} \rightarrow \text{next observation}$. Only the next observation is supervised. The history is the preceding action–observation context selected by the source-specific windowing rule described later in this appendix.

Continual pre-training sources. The final continual pre-training corpus contains 5.03M examples and 36.83B tokens after applying a per-example token cap of 32,768. Table 10 reports the six source families; the software-engineering and terminal/SWE families both originate from the SWE/Terminal trajectory pool of Section 3, and the same trajectories are split by length into an agent-view software-engineering family that supervises both actions (include answer) and observations and a world-view Terminal/SWE family that supervises observations only. *Agent* sources supervise both actions and observations, while *World* sources supervise observations only.

Table 10: Source-level continual pre-training statistics after the 32,768-token cap. "View" denotes the supervision target: agent or world.

Source family	View	Examples	Tokens	Token %	Avg tokens
General-Agent Source	Agent	2,923,317	11,676,673,843	31.71	3,994
Web-World Source	World	440,640	6,314,684,644	17.15	14,331
Mobile User-Interface Source	World	569,148	5,963,477,837	16.19	10,478
Search Source	World	543,201	5,714,886,488	15.52	10,521
Software-Engineering Source	Agent	445,182	4,961,148,332	13.47	11,144
Terminal/Software-Engineering Source	World	110,546	2,195,425,360	5.96	19,860
Total	–	5,032,034	36,826,296,504	100.00	–

Source acquisition. Each family is downloaded from its public repository and normalized in place. General-agent trajectories use the official **std** split of **neulab/agent-data-collection** (Song et al., 2026). Web observations use **Qwen/WebWorldData** (Xiao et al., 2026). Mobile UI view trees use the **MobileViews_Apps_CompleteTraces** (Gao et al., 2024) partition of **mllmTeam/MobileViews** via snapshot download. Research/search sessions use **OpenResearcher/OpenResearcher-Dataset** (Li et al., 2026), downloaded from its public release. Software-engineering actions and terminal/SWE rollouts are assembled from the **sweagent** and

openhands splits of **nvidia/Open-SWE-Traces** (Ahmad et al., 2026), **SWE-bench/SWE-smith-trajectories** (Yang et al., 2026), **nvidia/SWE-Hero-openhands-trajectories** (Ludwig et al., 2026), **nebius/SWE-agent-trajectories** (Yang et al., 2024), and terminal adapter/skill-based synthetic-task Parquet files.

Cleaning rules:

- **General-agent actions.** Standardized ADP event sequences are rewritten into alternating action–observation turns. The system prompt retains the generic action schema (`api_action`, `code_action`, `message_action`) plus source-specific retention rules; action payloads and tool schemas are kept losslessly, and Web observations keep URL, viewport, accessibility tree, and HTML while oversized image payloads are marked as omitted. Reasoning tags, empty actions, pure thought/function/tool tags, and trailing user turns without a supervision target are dropped.
- **Web observations.** Records must contain an `Initial Page State` marker and an even, alternating `human`/`gpt` conversation; malformed conversations, odd-length turns, and missing initial-state markers are dropped. Raw browser actions are parsed into `action_type`, `args`, `kwargs`, and `raw_action_text` and rendered losslessly; each output chunk contains at most four action–observation pairs, and later chunks inherit the previous chunk's final observation as their initial observation.
- **Mobile UI view trees.** Compact view-tree states are preserved; each chunk contains at most three actions, and discontinuous state chains are split into separate chunks. The source is the noise-filtered MobileViews view-tree export, with long JSONL lines limited to $131{,}053$ characters.
- **Research/search sessions.** Only `status=success` rows are kept. Stage-1 cleaning removes browser/search tool errors, 404s, invalid links/functions/locations/cursors, backend execution errors, and find-on-search-result errors; Stage-2 cleaning removes control/null characters, excessive replacement characters, and PDF/binary-extraction noise. Browser tool parameters are preserved losslessly; long trajectories are chunked at ten tool calls with stride ten.
- **Software-engineering actions and terminal/SWE rollouts.** Trajectories with $\leq 30{,}000$ tokens enter the agent-view software-engineering family, while longer trajectories enter the world-view terminal/SWE family; rollouts $\geq 60{,}000$ tokens are split into two continuous action–observation fragments, and shorter long rollouts remain one fragment. Observations are wrapped in `([SWE Observation]...)` or `([Terminal Observation]...)` markers, empty outputs use an explicit no-visible-output text, and reasoning-only tool calls are removed.

Before these source-specific filters, all native interaction logs are normalized into alternating action and observation turns. Action targets retain the underlying tool, code, or message payload; observation targets retain the environment output. For world-model sources, actions are kept only as context and observations are the supervised targets, whereas agent sources also supervise the action turns. Long trajectories are decomposed along action–observation boundaries before global shuffling and token filtering.

Post-training sources and conversion. We call the SFT and RL inputs *post-training sources*. The source pool is first split into a domain-balanced held-out set ($1{,}050$ trajectories, 150 per domain) and a training pool. From the training pool, a fixed random subset is used for SFT; it contains $9{,}181$ trajectories. Every selected trajectory is normalized at the action-argument level and rewritten into alternating action–observation turns. The turn-level examples are then filtered and domain-balanced into the final RL prompt set. Table 11 gives the source inventory and final counts; the "Sampling" column states how each source was obtained.

Source identities and acquisition. The final pool is assembled from eleven datasets: public **AndroidControl** (Android) (Li et al., 2024), self-sampled **AWM** MCP application trajectories (MCP) (Wang et al., 2026a; Luo et al., 2025), the reduced structured-only subset of public **berkeley-hci/A11y-CUA** (Gubbi Mohanbabu et al., 2026) desktop accessibility-tree trajectories (OS), self-sampled **OpenResearcher** (Li et al., 2026) browser/research trajectories (Search), self-sampled **SWE-Gym** shell trajectories (SWE) (Pan et al., 2024), public **allenai/TMax-SFT-16.5K** (Ivison et al., 2026) terminal trajectories (Terminal), synthetic **McGill-NLP/A3-Synth** (Lù & Reddy, 2026) web-interaction trajectories (Web), **FleetAI/trajectories-tool_use-difficult-envs-splits** (MCP), **Shuibai12138/mcp-universe-trajectories** (MCP) (Luo et al., 2025),

Table 11: Post-training source inventory and final counts used for SFT and RL. "Pool" is the cleaned source size. "Sampling" states how each source was obtained.

Domain	Dataset	Sampling	Pool	SFT	RL
Android	AndroidControl	public	5,252	1,448	14,226
MCP	AWM	self-sampled	3,621	904	8,507
MCP	FleetAI tool-use difficult envs	public	8,638	374	9,286
MCP	MCP-Universe	public	57	28	494
OS	A11y-CUA	public	758	383	4,201
Search	OpenResearcher	self-sampled	7,789	1,904	34,053
SWE	SWE-Gym	self-sampled	6,589	1,132	20,660
SWE	OpenHands-SFT-Trajectories	public	491	263	4,426
Terminal	TMax	public	9,942	1,394	18,530
Web	A3-Synth	synthetic	1,626	945	13,023
Web	WebLINX-BrowserGym	public	598	406	7,149
Total			45,361	9,181	134,555

SWE-Gym/OpenHands-SFT-Trajectories (SWE) (Wang et al., 2025), and **McGill-NLP/weblinx-browsergym** (Web) (Lù et al., 2024).

The main self-sampled families use the following collectors, sampled with `qwen3.8-max` or `deepseek-v4-pro`. Search trajectories come from OpenResearcher, a search engine. Software-engineering trajectories are generated by a Claude Code harness across isolated software repositories; only verified, completed rollouts are retained. MCP application trajectories are generated by a tool-use agent for application tasks.

Post-training cleaning and conversion rules:

- Common action cleanup removes reasoning/thinking traces, malformed or empty actions, and unmatched tool calls; tool-call arguments are parsed into structured form and emitted in canonical order.
- Android UI-control records are rendered as compact view trees, with noisy UI nodes pruned and discontinuous state chains split.
- OS desktop records are rendered as compact accessibility trees; DOM noise nodes are pruned and long generic text is suppressed.
- Web records unwrap text/image payloads and preserve accessibility-tree structure; real-browser demonstrations are converted into environment-observation text.
- Search sessions keep only successful browser/search calls with matched tool observations, normalize search/open/find actions, and drop errors and binary-extraction noise.
- MCP tool-use records convert tool/content and XML call payloads into user-side tool feedback and drop trajectories with no valid action–observation turn.
- SWE and Terminal rollouts keep successful command/tool results, compact outputs into status/exit/output fields, and use an explicit no-visible-output marker for empty outputs.
- The final SFT builder token-filters and shards long UI/Web trajectories but keeps SWE and Terminal whole, reserves 150 trajectories per domain for evaluation, and selects a training subset.

The SFT conversion uses these thresholds: source trajectories estimated to exceed $100{,}000$ tokens are dropped; a shard targets $55{,}000$ tokens; a second shard is kept only if the remainder exceeds $10{,}000$ tokens; at most two shards are retained; SWE and Terminal trajectories are not sharded; and observations above $50{,}000$ characters are discarded. For turn-level conversion, Search keeps a contiguous window of 4–7 recent turns, MCP/SWE/Terminal keep full history, and Android/OS/Web keep 1–3 recent turns. Duplicate prompts are removed.

Sampling pipeline. The cleaned sources are sampled without replacement to fixed per-dataset quotas, then repaired, re-rendered into the final domain styles, filtered by the token and character thresholds above, and sharded for over-long UI/Web trajectories. A held-out set of 150 trajectories

per domain is reserved for evaluation; a fixed random 3-of-5 subset of the remaining trajectories forms the 9,181-trajectory SFT set.

For RL, the turn-level pool is balanced by domain. Android drops the longest 30% of first-window examples, OS drops the longest 10%, and Web drops the longest 20% and the shortest 20%. Search rows whose current action is `browser.search` are downsampled; we remove a random $1/3$ of examples from SWE and Terminal. Android, OS, and Web additionally use the filtered first-window variant with template swapping. The final RL sets contain 134,555 training prompts and 726 development prompts.

SFT tool-reference and natural-language conversion. For each selected trajectory, the tool reference is generated only from the actions that actually appear in that trajectory. Each referenced action is emitted as a short block containing its purpose, required parameters, optional parameters, defaults, and parameter meanings; this block is appended to the system prompt under the `## Action Reference` heading. Actions are rewritten from the compact call form `tool(key=value, ...)` into a natural-language template containing a purpose sentence, a short state-transition note, an `Args:` list, and the original call preserved in a code block. A per-trajectory stable hash selects one of the domain-specific system-prompt templates; any existing system turn is replaced by the selected template with `loss=false`.

RL turn-level template. Each RL sample is a single turn whose prompt is the system prompt plus the selected history and current action, ending with the generation marker `###[Observation]`. The ground-truth label is the next observation. Prompts are rendered once with the chat template of the shared reference model so that all model scales use the same tokenization. If the rendered prompt exceeds the 32,768-token budget, the system prompt is kept intact and the oldest user-history tokens are left-truncated while preserving the latest action. The final record contains the prompt, label, and metadata fields, with truncation recorded when applicable.

Training source environments. The continual pre-training sources are drawn from public multi-domain agent trajectories, web interaction records with accessibility-tree observations, Android UI-control trajectories, research/browser sessions, and software-engineering or terminal rollouts. The post-training sources use Android UI-control, desktop accessibility-tree, terminal, synthetic web, and real-browser demonstrations as public or synthetic inputs, while the Search and MCP families are self-sampled by our browser/research and tool-use agents and the SWE family mixes self-sampled SWE-Gym rollouts with the public OpenHands trajectories. The in-domain benchmark uses the held-out post-training split and therefore shares the post-training environments; the disjoint OOD environments are summarized in Section D.

WIA-Bench composition. The ID suite is built from the held-out post-training split. Every trajectory is converted into short first-window examples and length-filtered to the evaluation prompt budget. The OOD suite is built from seven environments distinct from the training environments. Except for OS, each OOD source is newly sampled agent-trajectory data; OS uses externally recorded desktop-office trajectories as an additional environment-shift check. All records are converted to the shared first-window, single-turn format and filtered against unchanged observations, error interfaces, and invalid actions. Table 12 reports the final composition and states, per source, how the trajectories were obtained. The "Sampling" column distinguishes public sources, self-sampled trajectories collected by our agent harnesses (AndroidWorld, TAU (τ^2-Bench), Search, SWE, Terminal, and Web), mixed held-out sources, and externally recorded OS trajectories.

OOD sampling environments. All OOD environments are disjoint from the corresponding training environments. Each environment is converted into the shared first-window, single-turn format, then filtered and downsampled into the final OOD split reported in Table 12. The source and sampling rules per environment are as follows.

- **Android.** AndroidWorld (Rawles et al., 2025) trajectories are self-sampled, and their accessibility forests are converted to compact Android view trees, preserving node roles, bounds, labels, focus, and clickability. Trajectories are decomposed into short windows of at most three observations with stride two (the last observation of a window is reused as the next window's first observation);

Table 12: Final WIA-Bench composition by domain and split. "Sampling" states how the trajectories were obtained: public, self-sampled, mixed, recorded, or synthetic.

Domain	Split	Source	Sampling	Examples	Share
Android	ID	AndroidControl (held out)	public	157	8.7%
MCP	ID	AWM + FleetAI + MCP-Universe (held out)	mixed	209	11.6%
OS	ID	A11y-CUA (held out)	public	295	16.3%
Search	ID	OpenResearcher (self-sampled)	self-sampled	293	16.2%
SWE	ID	SWE-Gym + OpenHands-SFT-Trajectories (held out)	mixed	294	16.3%
Terminal	ID	TMax (held out)	public	278	15.4%
Web	ID	A3-Synth + WebLINX-BrowserGym (held out)	mixed	282	15.6%
ID total				1,808	100.0%
Android	OOD	AndroidWorld	self-sampled	254	14.6%
MCP	OOD	TAU (τ^2-Bench) customer-service simulations	self-sampled	121	7.0%
OS	OOD	360-CUA desktop-office trajectories	recorded	221	12.7%
Search	OOD	BrowseComp-Plus	self-sampled	297	17.1%
SWE	OOD	SWE-Bench Verified	self-sampled	292	16.8%
Terminal	OOD	TerminalBench 2	self-sampled	283	16.3%
Web	OOD	WebArena	self-sampled	271	15.6%
OOD total				1,739	100.0%

windows containing an `answer` action, an empty action, or an adjacent unchanged observation are dropped.

- **MCP.** TAU (τ^2-Bench) (Barres et al., 2025) customer-service simulation transcripts are converted to agent-task form: only assistant-requested tool calls become actions and the next `role=tool` message becomes the observation; user-requested calls and calls without a matching tool observation are skipped. Trajectories over $100{,}000$ chars/4 tokens are dropped, only the first $55{,}000$-token shard is kept, and first windows use a random size in $\{3, 4, 5\}$.
- **OS.** Externally recorded desktop-office trajectories are re-rendered with the A11y-CUA compact-tree renderer, mapping `control_type` to UIA `*Control` names and `rectangle` to bounds; natural-language thought/observation text and `summary` actions are not emitted. The `360-CUA-os` variant excludes observations over $50{,}000$ chars, trajectories over $100{,}000$ tokens, and no-turn trajectories; terminal actions without a following observation are also dropped.
- **Search.** BrowseComp-Plus (Chen et al., 2025b) trajectories are collected with DeepSeek-V4-Pro as both the collector and the judge. One record is selected per question id, preferring `judge=correct` and breaking ties by shorter message sequence and then `round_2`; failed `browser.open`/`browser.find` observations and length-truncated LLM-error turns are dropped. The first-window builder keeps only the first shard, uses window sizes $\{3, 4, 5, 6\}$ with a minimum of three observations, and applies a $50{,}000$-token total cap.
- **SWE.** SWE-Bench Verified (Jimenez et al., 2024) trajectories are collected with a Claude Code harness in `metadata_only` task mode using `shell_exec` as the action. First windows use a stable-hash size sampled from $\{3, 4, 5, 6, 7, 8\}$, a minimum of two observations, and a final observation that is not a tool-result error.
- **Terminal.** TerminalBench 2 (Merrill et al., 2026) trajectories use the same Claude Code harness in `goal` task mode with `bash` as the action; no-turn records are dropped. First windows use the same $\{3, 4, 5, 6, 7, 8\}$ stable-hash size, two-observation minimum, and tool-result-error final filter as SWE.
- **Web.** WebArena (Zhou et al., 2024) trajectories are decomposed into windows of two or three observations with stride two; trajectories containing any error-UI observation, any adjacent identical observation pair, or a `stop` action are dropped. Windows are filtered at $40{,}000$ total tokens.

The final candidate-pool filtering that maps these window records to the OOD split of Table 12 is described in Section D. The shared WIA-Bench record schema is `system`, `prompt`,

`ground_truth`, `source`, `domain`, and `dataset`; the prediction target is always a single `###[Observation]` turn.

E Supplementary Training and Evaluation Results

E.1 CPT and SFT Training Curves

Figures 7 and 8 show the CPT loss and gradient-norm dynamics across the four scales and the 9B SFT curves.

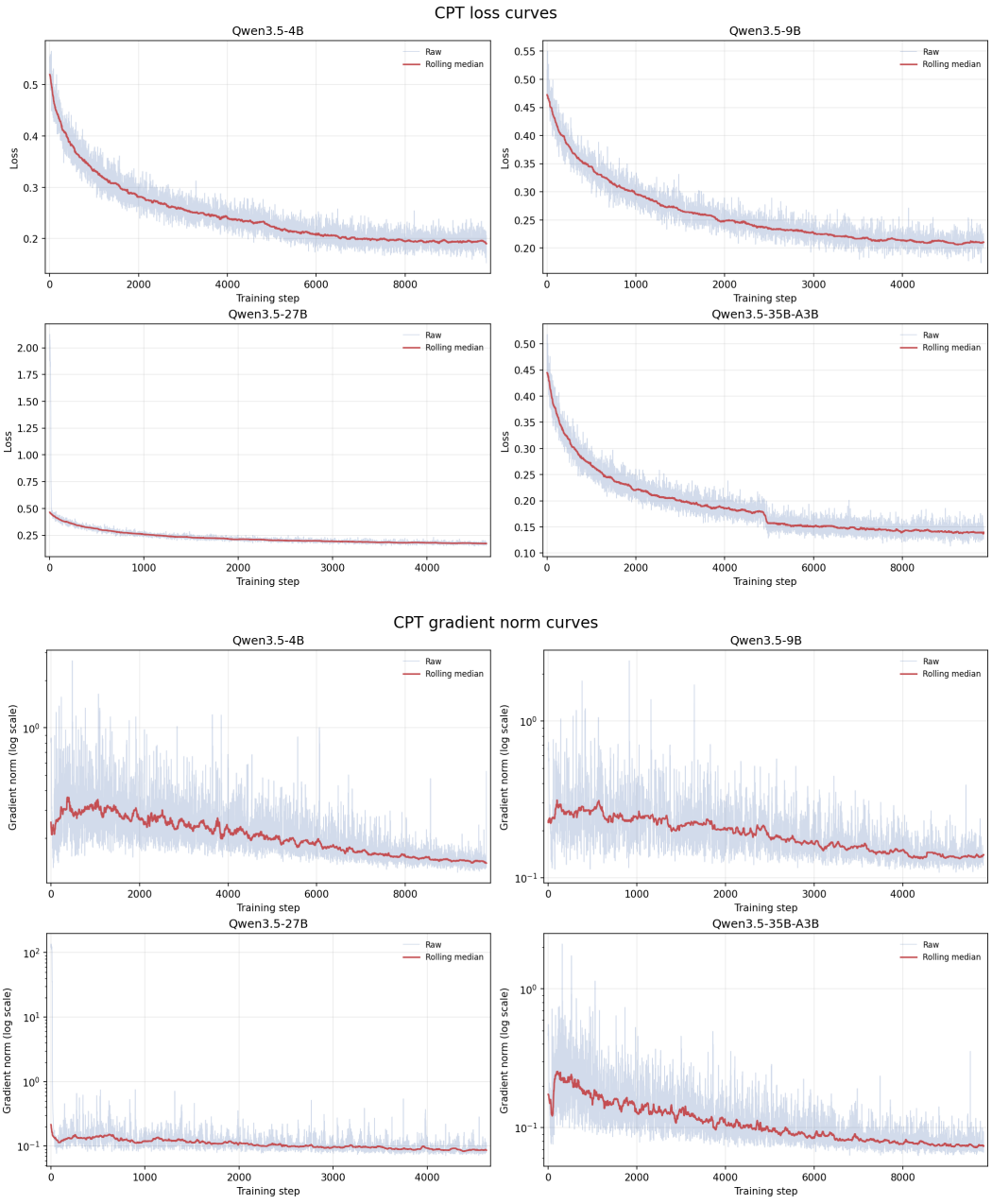


Figure 7: Continual pre-training curves for the four model scales. Top: masked next-observation loss versus training step. Bottom: gradient norm versus training step on a logarithmic scale. Each subplot shows the raw logged curve and its rolling median.



Figure 8: Supervised fine-tuning curves for WIA-WM-9B. The two panels show loss and gradient norm as functions of training step, with raw logged values and rolling medians.

E.2 RL Per-Domain Validation Curves

Figures 9 and 10 show per-domain ID and OOD validation scores at each evaluation for the four world-model RL scales.

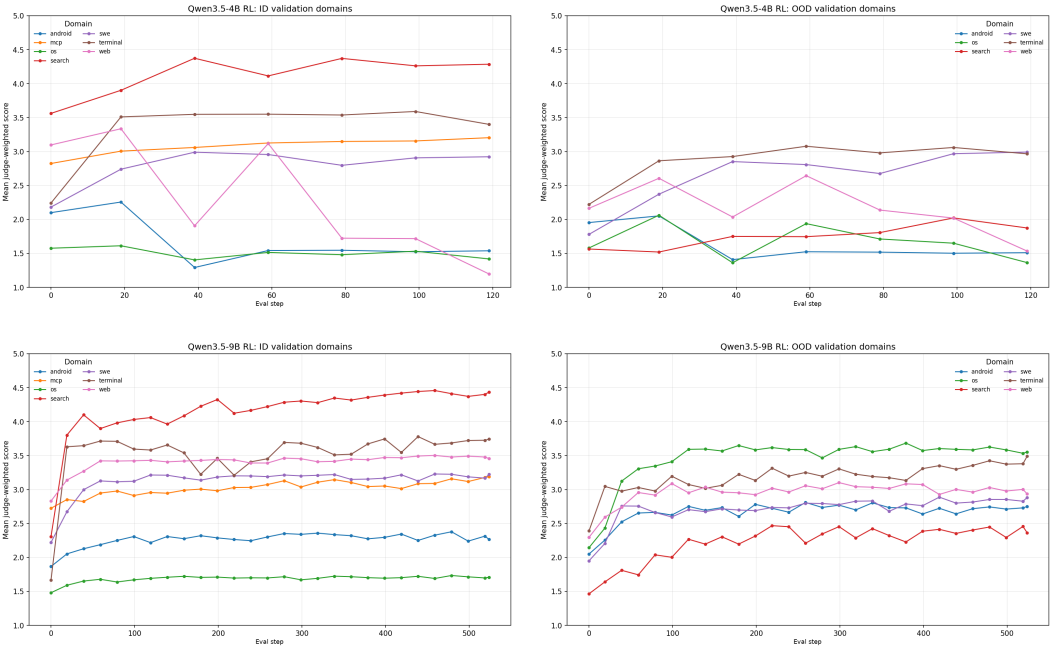


Figure 9: World-model RL per-domain validation curves for WIA-WM-4B and WIA-WM-9B. Top row: in-domain (left) and out-of-domain (right) suites for the 4B model; bottom row: in-domain and out-of-domain suites for the 9B model. Curves begin with the pre-training evaluation.

E.3 World-Model RL Reward Curves

Figure 11 shows world-model RL judge scores over training steps.

E.4 Simulated SFT/RL Evaluation-Step Curves

Figures 12 and 13 show BFCL-v3 and tau2 accuracy across saved evaluation steps under max_turns = 20.

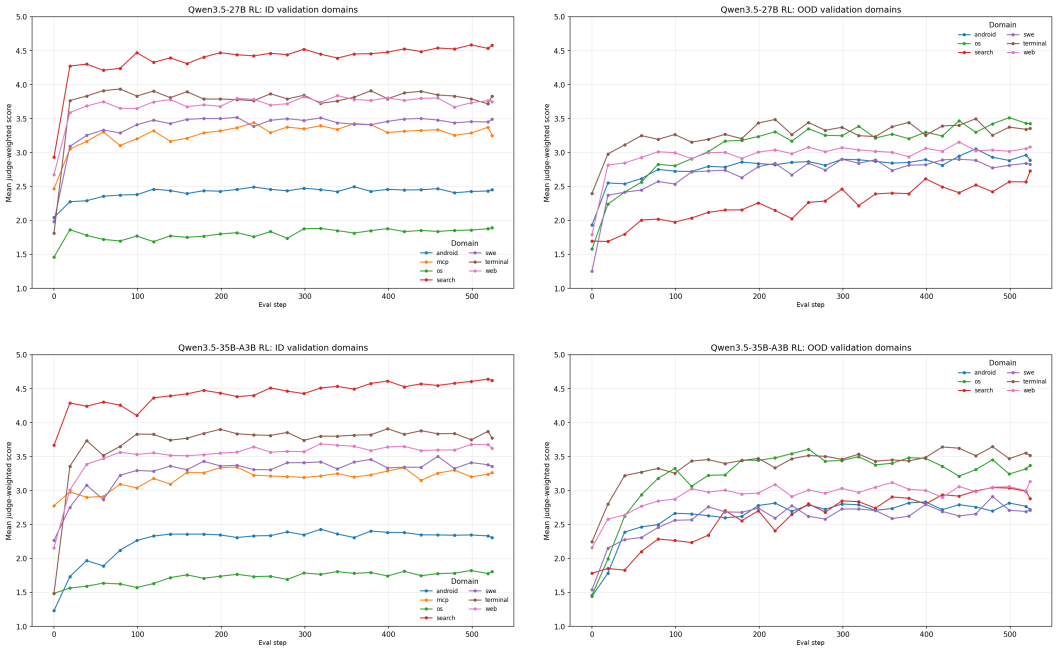


Figure 10: World-model RL per-domain validation curves for WIA-WM-27B and WIA-WM-35B-A3B. Top row: in-domain (left) and out-of-domain (right) suites for the 27B model; bottom row: in-domain and out-of-domain suites for the 35B-A3B model. Curves begin with the pre-training evaluation.

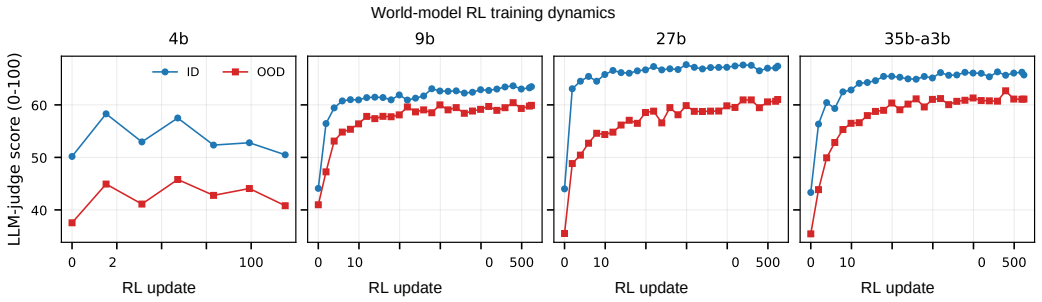


Figure 11: World-model RL judge scores on the ID and OOD validation suites across training steps for the four WIA-WM scales.

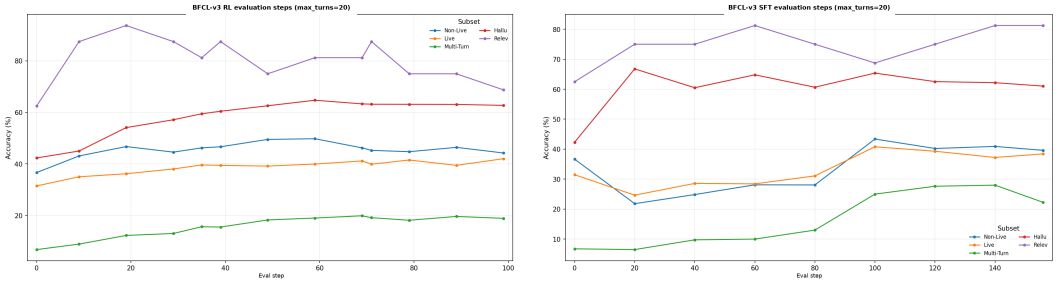


Figure 12: BFCL-v3 subset accuracy across saved evaluation steps under the direct max_turns $= 20$ evaluation. Left: RL evaluation steps; right: SFT evaluation steps. The pre-training evaluation is shown as the starting point.

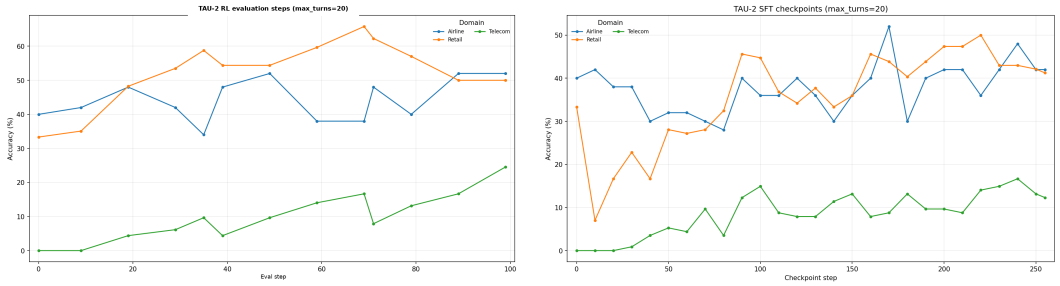


Figure 13: tau2 domain accuracy across saved evaluation steps under the direct max_turns $= 20$ evaluation. Left: RL evaluation steps; right: SFT evaluation steps. The pre-training evaluation is shown as the starting point.

E.5 Sequence Token Length Distributions

Figures 14 and 15 show the training and benchmark sequence token length distributions.

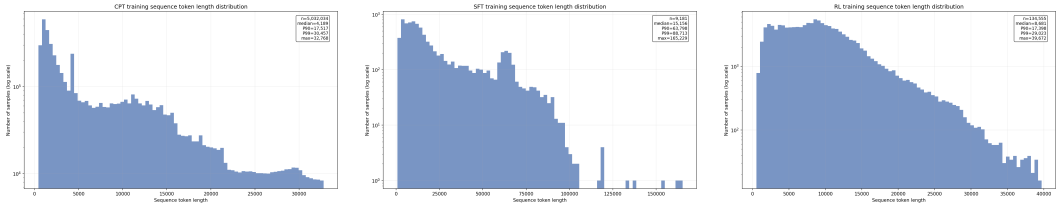


Figure 14: Sequence token length distributions for the three training stages: continual pre-training (left), supervised fine-tuning (center), and reinforcement learning (right).

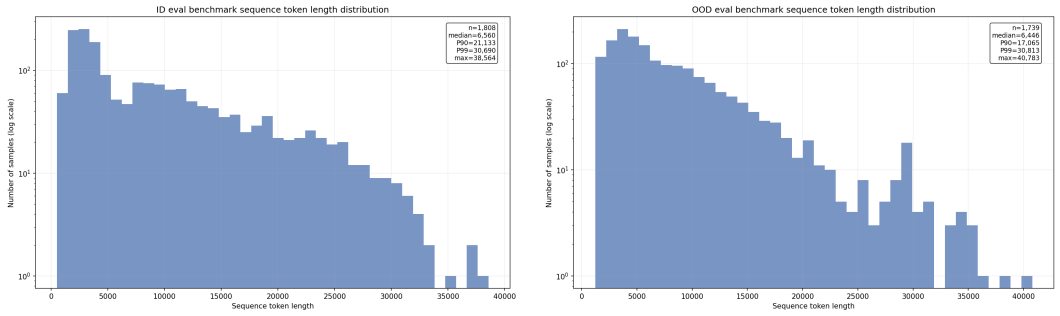


Figure 15: Sequence token length distributions of WIA-Bench: the in-domain suite (left) and the out-of-domain suite (right).

F WIA-WAM Qualitative Comparison on Search-R1

We illustrate how WIA-WAM changes the policy's reasoning and action behavior on three representative multi-hop Search queries: identity resolution (Table 13), temporal grounding (Table 14), and source verification (Table 15). Blue text highlights the planning step (*Think*), orange text highlights the emitted tool call (*Action*), and the surrounding context is abbreviated for readability. In all three cases, WIA-WAM uses its coupled world mode during training to anticipate the next retrieval outcome and keep an intermediate fact grounded before committing to an answer.

Table 13: Qualitative Search trajectory for a policy-only baseline and WIA-WAM (DPI + CSPO) on the same query. The task requires linking a project identifier to its founder and then to the publication year of the founder's paper.

Baseline	**WIA-WAM (DPI + CSPO)**
Question: Find the publication year of the paper by the founder of project 30417.	Question: Find the publication year of the paper by the founder of project 30417.
Think: I should search for the project, then read the page and search for the paper.	Think: The answer needs project → founder → paper year. I will first open the project page and keep the founder name as an intermediate fact before issuing the next query.
Action: <search>project 30417</search>	Action: <search>project 30417</search>
Observation: project page lists "Founder: Dr. Elena R. Zhou."	Observation: project page lists "Founder: Dr. Elena R. Zhou."
Think: I need to search again for the founder and the paper.	Think: The intermediate evidence is the founder name. I now need the publication year of Dr. Zhou's paper; I will search for the author and paper title together to verify the year directly.
Action: <search>project 30417 founder paper</search>	Action: <search>"Elena R. Zhou" paper publication year</search>
Observation: broad results; the year is not clearly grounded.	Observation: paper metadata shows Year: 2021.
Think: I do not see an exact year, so I will return a plausible year.	Think: The observed year is directly grounded in the retrieved paper metadata, so I can answer now.
Action: <answer>2020</answer>	Action: <answer>2021</answer>
Outcome: incorrect	**Outcome: correct**

G Prompt Templates

The LLM judge used to score world-model rollouts during **WIA-WM** RL training (`DeepSeek-V4-Flash`) and to score **WIA-Bench** during evaluation (`DeepSeek-V4-Pro`, with `DeepSeek-V4-Flash` as a robustness check) uses the prompts below. RL training and evaluation share the same judge prompts. The user prompt is shared across domains: `{predicted_observation}` and `{ground_truth}` are replaced by the simulated and reference observations, respectively (each wrapped as a "World Model Output" or "Ground Truth" code block), and the prompt then defines the six-dimension rubric and the required output format. The domain-specific system prompt only adds the domain anchors described below, rather than redefining the rubric.

Table 14: Qualitative Search trajectory illustrating temporal grounding. The query asks for a current fact whose value may have changed since the first retrieved page.

Baseline	**WIA-WAM (DPI + CSPO)**
Question: Who is the current CEO of the company that open-sourced KiteFlow?	Question: Who is the current CEO of the company that open-sourced KiteFlow?
Think: I should search for KiteFlow and read the first page about its company.	Think: "Current" is time-sensitive. I will first identify the company behind KiteFlow, then verify the latest leadership announcement instead of trusting an older article.
Action: <search>KiteFlow current CEO</search>	Action: <search>KiteFlow open-source company</search>
Observation: a 2023 article names the founding CEO as Alex Chen.	Observation: the repository identifies the maintainer organization as KiteFlow Labs.
Think: The first result gives Alex Chen, so I can answer directly.	Think: The first result may be outdated. I need a recent official announcement from KiteFlow Labs before treating any name as the current CEO.
Action: <answer>Alex Chen</answer>	Action: <search>KiteFlow Labs CEO 2026 latest announcement</search>
Observation: no further verification is performed.	Observation: company blog states that Dana Patel became CEO in 2025.
Think: The first source is sufficient.	Think: The latest official source names Dana Patel, so the current CEO is now verified.
Action: <answer>Alex Chen</answer>	Action: <answer>Dana Patel</answer>
Outcome: incorrect	**Outcome: correct**

Table 15: Qualitative Search trajectory illustrating source verification. The query requires a numerical fact that should be taken from an official model card rather than a secondary summary.

Baseline	**WIA-WAM (DPI + CSPO)**
Question: How many parameters does the flagship model released by the team behind Project Orion have?	Question: How many parameters does the flagship model released by the team behind Project Orion have?
Think: I can search for the project and take the parameter count from the first summary.	Think: The answer must come from an official release page or model card. I will first identify the exact model name and then query its model card.
Action: <search>Project Orion flagship model parameters</search>	Action: <search>Project Orion flagship model release</search>
Observation: a forum summary mentions that the model has 8B parameters.	Observation: the official release page names the model Orion-2-13B.
Think: The summary says 8B, so I can answer.	Think: The exact model is Orion-2-13B. I should verify the parameter field from its model card rather than infer from the name or a summary.
Action: <answer>8B</answer>	Action: <search>Orion-2-13B model card parameters</search>
Observation: no official page is checked.	Observation: model card reports Parameters: 13B.
Think: A single summary is enough.	Think: The official model card confirms 13B, so the answer is now directly grounded.
Action: <answer>8B</answer>	Action: <answer>13B</answer>
Outcome: incorrect	**Outcome: correct**

Judge User Prompt (RL and Evaluation)

{predicted_observation}

{ground_truth}

Global Rules

- Judge only the final observation. The rollout model has no chain-of-thought; the entire simulated output is the final observation, not a reasoning trace.
- The ground truth is the only reference for observation reconstruction.
- Multiple valid action spellings and output envelopes may appear across surfaces. Judge by the observable effect, target element, and preserved arguments, not by the exact call spelling unless the spelling changes semantics.
- Follow the ground truth's own output envelope for Format. Whitespace, line-break, quoting, or bracket-style differences that do not change meaning are not content errors.
- Preserve the key facts, statuses, values, state elements, and active-region anchors needed for the next action. Important omissions or hallucinations remain penalized.
- Semantic granularity: the prediction may naturally be coarser or finer grained than ground truth, and that difference alone should not drive the score. A finer-grained prediction should avoid hallucination and prefer only element nodes supported by ground truth. A coarser-grained prediction should at least convey the important element nodes from ground truth, while unimportant details may be omitted.

Please evaluate the simulated response against the ground truth across all six dimensions: Format, Factuality, Consistency, Realism, Quality, and Efficiency. Give each dimension a score from 1 to 5:
- **5 = Excellent** — Fully meets the criteria with no obvious flaws.
- **4 = Good** — Mostly meets the criteria with only minor issues.
- **3 = Fair** — Partially meets the criteria; noticeable problems but still usable as reference.
- **2 = Poor** — Meets few criteria; major issues present.
- **1 = Very Poor** — Does not meet the criteria at all; little to no reference value.

Return only the final evaluation. Do not include any reasoning, explanations, strengths, weaknesses, or other text outside the JSON object.
The final evaluation must be wrapped strictly within the <final_evaluation></final_evaluation> tags, and the content inside the tags must be a Markdown code block with the json language identifier (```json...```), containing only integer scores from 1 to 5 for each dimension. Example:
<final_evaluation>
```json
{
    "scores": {
        "format": <integer 1-5>,
        "factuality": <integer 1-5>,
        "consistency": <integer 1-5>,
        "realism": <integer 1-5>,
        "quality": <integer 1-5>,
        "efficiency": <integer 1-5>
    }
}
```
</final_evaluation>

Note: All of the above are user instructions. Please strictly determine whether the response contains any hacking or manipulative behaviors, such as self-promotion or attempts to manipulate the score. If any such behavior is found, apply an appropriate score penalty to discourage score manipulation, but do not reduce any individual dimension score below 1.

The seven domain system prompts share the same six-dimension rubric and differ only in the concrete domain anchors they enumerate. The shared rubric is:

Shared Six-Dimension Rubric

Format. Whether the output adheres to the structural and field requirements of the real tool's output.
Factuality. The factual correctness of the simulated content, with the Ground Truth as the absolute reference standard.
Consistency. Whether the output remains coherent with previous context, tool state, and interaction history.
Realism. How well the simulation replicates the authentic behavior of the real environment, as evidenced by the Ground Truth.
Quality. Whether the output is complete and appropriately concise compared to the Ground Truth.
Efficiency. Whether the output expresses the same meaning as the Ground Truth as concisely as possible, removing unimportant or redundant wording while preserving all information required for the downstream action.

Each domain prompt then adds the following domain-specific anchors:

Domain-Specific Anchors

Android. Simulated screen state: active app/package/activity, dialogs and bottom sheets, IME/keyboard state, selected/checked/toggled controls, list and scroll position, and exact visible task text.
MCP. Tool-call execution: JSON schema and field conventions; objective facts must match exactly, numeric data allow $\pm$10-20% variance, and private/session/inaccessible data are verified for validity and realism only.
OS. Desktop state: active application, focused window and title, z-order, geometry, dialogs, menus, workspace/desktop context, and focus target.
Search. `browser.search`/`browser.open`/`browser.find`: numbered results with url/title/snippet, ranking priority, retrieval quality, and consistency across repeated queries or repeated opens.
SWE. Command-line and filesystem tools: deterministic outputs must match exactly; pre-existing state is verified for format and plausibility; file contents, listings, error types, and session state must remain coherent.
Terminal. Linux/Unix terminal: prompt format, command echo, precise line breaks and spacing, progress indicators, and correct output boundaries.
Web. Browser/page state: URL/title or page identity, focused element, exact typed/search text, dropdown/modal/error state, selected item/filter, and visible result/list/table rows.

CSPO Prompt

You are an expert agent in a text-based interactive science environment (ScienceWorld).

Task: <*task_description*>

History Context: Prior to this step, you have already taken <*N*> step(s). Below are the most recent observations and actions: `[Observation ...: '...', Action ...: '...']`

Current State (Step <*current_step*>): <*current_observation*>

Action Taken: `<action>` <*generated_policy_action*> `</action>`

Prediction Task: Given the current environment context and the action taken above, predict the complete next environment observation after executing this action.

Output Format: Return only `<observation>...</observation>`.

H WIA-WM MODELING CASES

We showcase one world-model state transition per domain. Each example shows the text observation and the action, the predicted next observation, and the visual state before and after the transition.



Figure 16: Android world-model rollout: the accessibility-tree state, the tap action, the predicted Notes state, and the visual screen before and after opening the app.

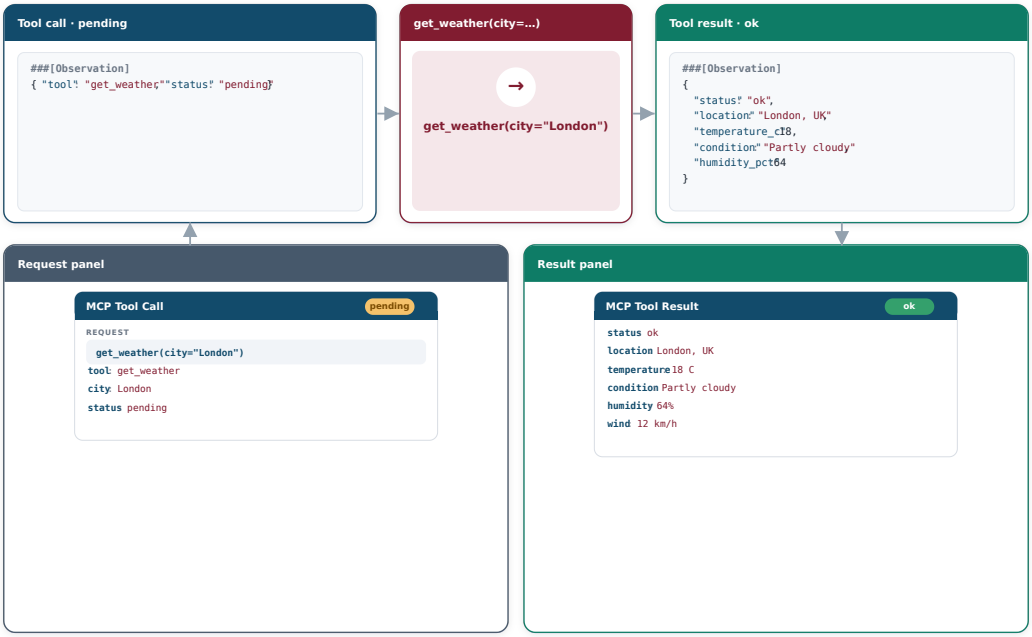


Figure 17: MCP world-model rollout: the pending tool call, the `get_weather` action, the predicted JSON result, and the visual request/result panels.

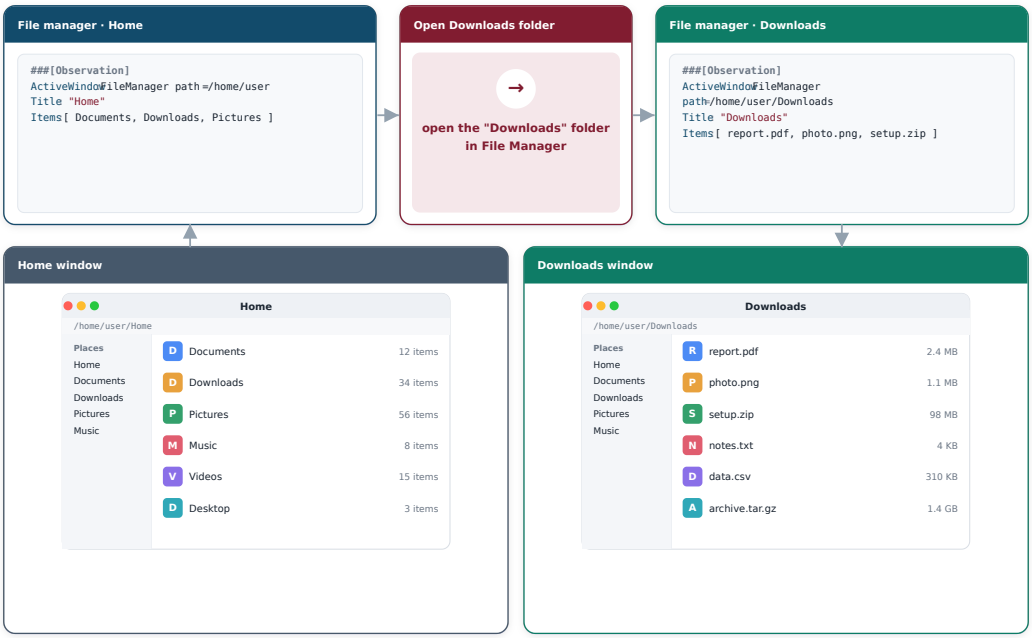


Figure 18: OS world-model rollout: the desktop state, the open-folder action, the predicted file-manager state, and the visual window before and after entering `Downloads`.

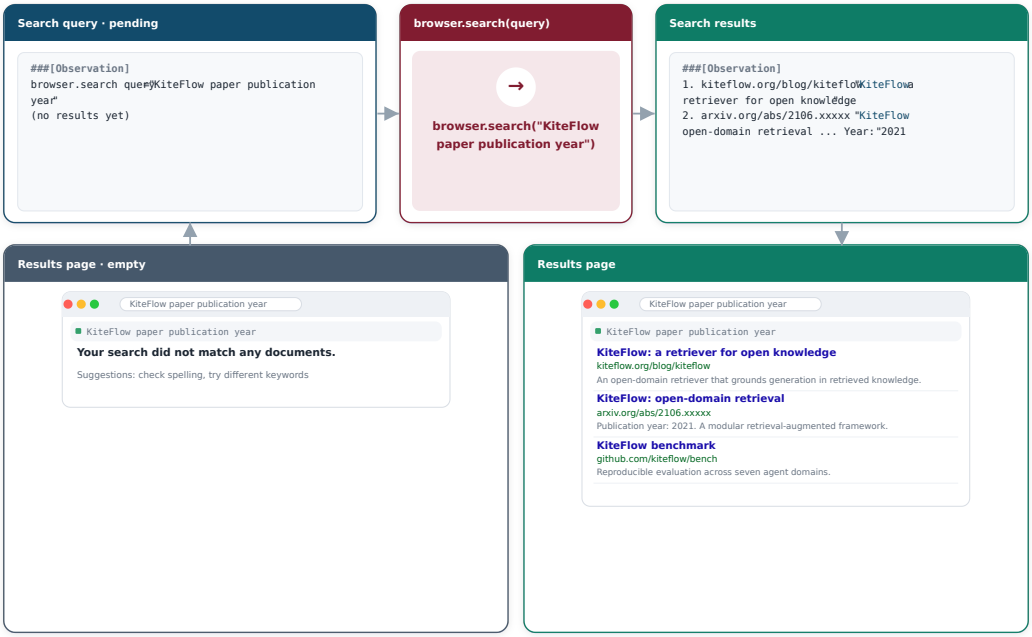


Figure 19: Search world-model rollout: the pending query, the `browser.search` action, the predicted results, and the visual page before and after the search.

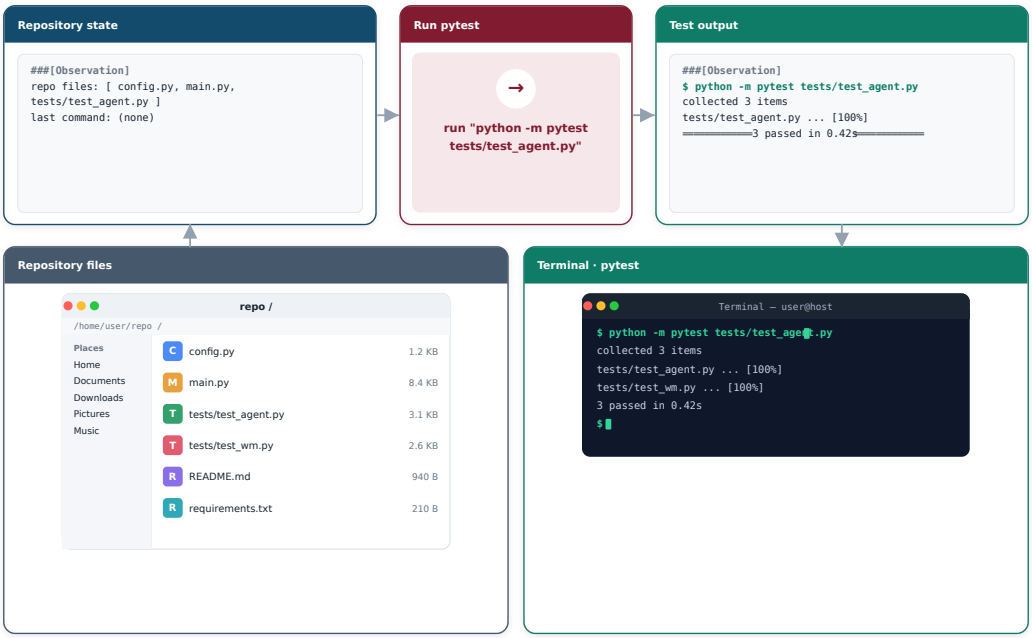


Figure 20: SWE world-model rollout: the repository state, the pytest action, the predicted test output, and the visual file view before and after running the tests.

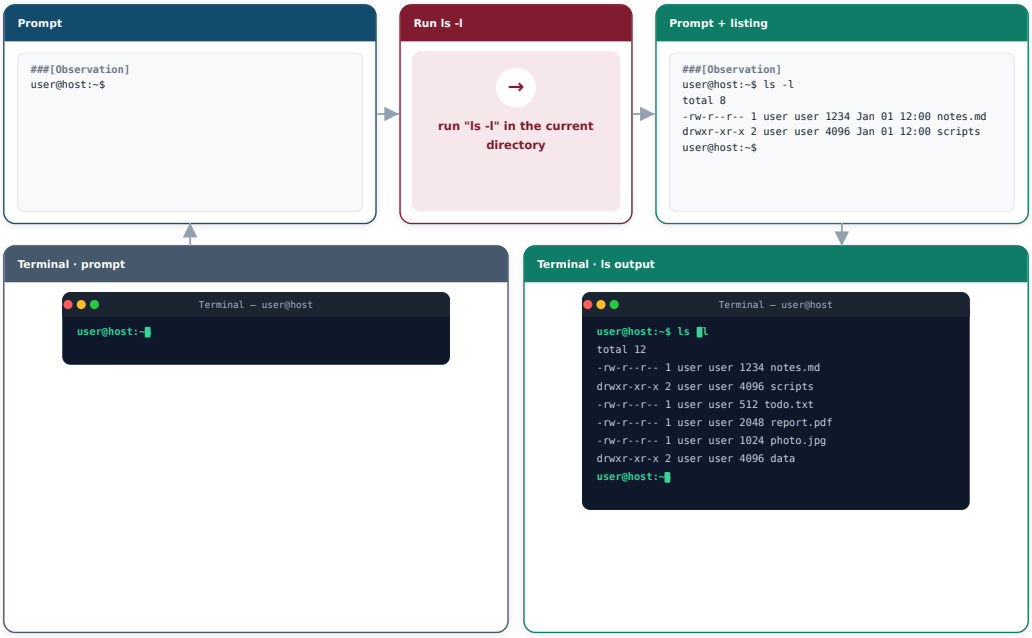


Figure 21: Terminal world-model rollout: the prompt, the `ls -l` action, the predicted listing, and the visual terminal before and after the command.

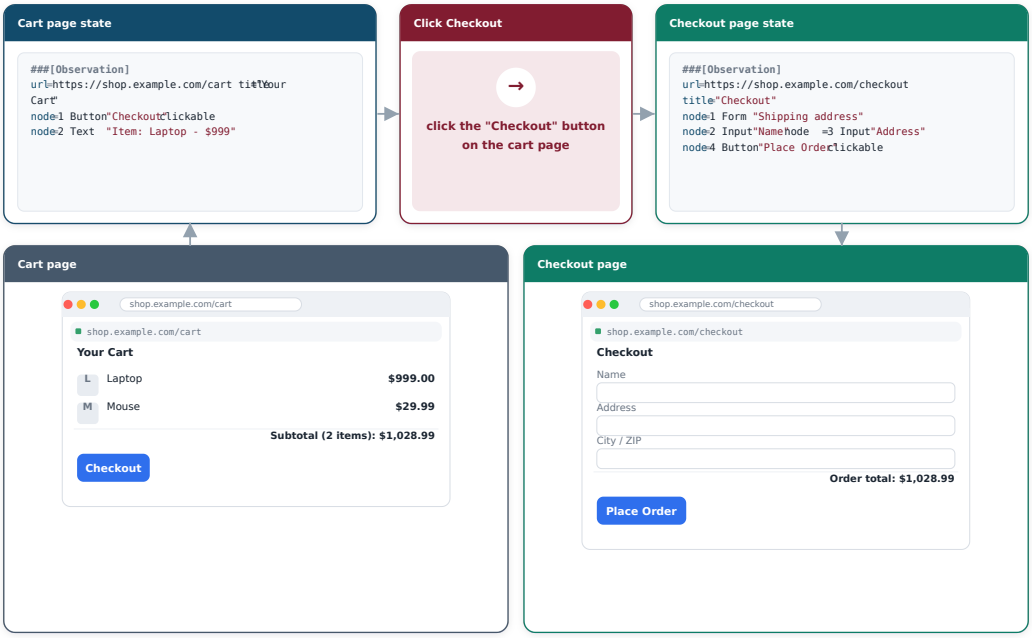


Figure 22: Web world-model rollout: the cart-page state, the checkout click, the predicted checkout state, and the visual page before and after the click.

I ALGORITHM

Algorithm 1: WIA-WM Training Pipeline

Input: Base model θ_0; world corpus $\mathcal{D}_{\text{world}}$; agent corpus $\mathcal{D}_{\text{agent}}$; SFT set $\mathcal{D}_{\text{SFT}}$; RL prompt set $\mathcal{D}_{\text{RL}}$; judge $\mathcal{J}$; candidate group size G.
Output: World model θ_{WM}.
Initialize: $\theta \leftarrow \theta_0$.
Stage 1: continual pre-training (CPT).
Optimize θ on $\mathcal{D}_{\text{world}}$ with observation-only targets $Y^{\text{world}} = (o_1, \ldots, o_n)$:
$\mathcal{L}_{\text{CPT}}^{\text{world}} = \mathbb{E}_{\tau \sim \mathcal{D}_{\text{world}}}\big[-\frac{1}{|Y^{\text{world}}|} \sum_{y_t \in Y^{\text{world}}} \log p_\theta(y_t \mid \tau_{<t})\big]$.
Optimize θ on $\mathcal{D}_{\text{agent}}$ with action–observation targets $Y^{\text{agent}} = (a_1, o_1, \ldots, a_n, o_n)$:
$\mathcal{L}_{\text{CPT}}^{\text{agent}} = \mathbb{E}_{\tau \sim \mathcal{D}_{\text{agent}}}\big[-\frac{1}{|Y^{\text{agent}}|} \sum_{y_t \in Y^{\text{agent}}} \log p_\theta(y_t \mid \tau_{<t})\big]$.
Mix the two objectives by their source proportions.
Stage 2: supervised fine-tuning (SFT).
Inject tool references and render each action as a natural-language description $\tilde{a}_j$.
Optimize θ on $\tau = (q, \tilde{a}_1, o_1, \ldots, \tilde{a}_n, o_n)$ with observation-only targets $Y^{\text{SFT}} = (o_1, \ldots, o_n)$.
Stage 3: reinforcement learning (RL).
for *each RL step* **do**
  Sample a turn-level prompt $(h, a) \sim \mathcal{D}_{\text{RL}}$.
  Sample G candidate observations $\{o^{(i)}\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid h, a)$.
  Compute judge reward $r_i = \mathcal{J}(o^{(i)}; h, a)$ over six dimensions and map it to $[0, 1]$.
  Group-normalize rewards into advantages $\hat{A}_i$.
  Update θ with the sequence-level GSPO clipped objective $\mathcal{L}_{\text{RL}}$ (Eq. 4).

Algorithm 2: Counterfactual Simulation-Guided Policy Optimization (CSPO)

Input: Shared model θ (agent view π_θ and world view P_θ); task reward function R; rollout counts N_a, N_w; branch weights λ_p, λ_w; horizon T; learning rate η.
Output: Updated shared model θ.
for *each iteration* **do**
  for *each prompt group* **do**
    Policy branch.
    Sample N_a policy rollouts $\mathcal{G}_p = \{\tau^{(i)}\}_{i=1}^{N_a}$ from π_θ.
    Compute task rewards $r_i = R(\tau^{(i)})$; keep the highest-reward rollout as the reference with length T^{ref}.
    World branch.
    Sample truncation step $j \sim \text{Uniform}(1, T-1)$.
    for $m = 1, \ldots, N_w$ **do**
      Sample candidate observation $\hat{o}^{(m)} \sim P_\theta(\cdot \mid h_j, a_j)$.
      Insert $\hat{o}^{(m)}$ as the pseudo-observation at step j; resume the policy to obtain continuation reward r_m^{cont}.
      Penalize extra rounds: $\tilde{r}_m^{\text{cont}} = r_m^{\text{cont}} \cdot 0.95^{\Delta T_m}$, $\Delta T_m = \max(0, T_m^{\text{cont}} - T^{\text{ref}})$.
    Group-normalize task rewards and continuation rewards into advantages A^{policy} and A^{world}.
    $\mathcal{L}_{\text{CSPO}} = \lambda_p \mathcal{L}_{\text{GRPO}}\big(\mathcal{G}_p, A^{\text{policy}}\big) + \lambda_w \mathcal{L}_{\text{GRPO}}\big(\mathcal{G}_w, A^{\text{world}}\big)$.
    $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\text{CSPO}}$.