Policy and World Modeling Co-Training for Language Agents

Ning Lu[1,2,*], Baijiong Lin[3,*], Shengcai Liu[1,†], Jiahao Wu[1,4], Haoze Lv[1], Yanbin Wei[2,5], Lingting Zhu[6], Shengju Qian[7], Xin Wang[7], Yingcong Chen[2,3], Qi Wang[5], Ke Tang[1]
[1]Guangdong Provincial Key Laboratory of Brain-Inspired Intelligent Computation,
Department of Computer Science and Engineering, Southern University of Science and Technology
[2]The Hong Kong University of Science and Technology
[3]The Hong Kong University of Science and Technology (Guangzhou)
[4]The Hong Kong Polytechnic University [5]Southern University of Science and Technology
[6]The University of Hong Kong [7]LIGHTSPEED

Abstract

Reinforcement learning (RL) improves large language model (LLM) agents by teaching them which actions lead to high rewards, but provides little supervision on what those actions do to the environment. World modeling (WM) can fill this gap, yet existing approaches often require separate simulators, extra training stages, or additional inference-time computation. We observe that on-policy RL rollouts already contain the needed signal: each transition pairs an action with its resulting next observation. Based on this observation, we propose **PaW**, a **P**olicy **a**nd **W**orld modeling co-training framework that adds auxiliary WM supervision to the same policy during RL, without changing the inference paradigm. To make auxiliary WM supervision informative and stable, PaW introduces three components: action-entropy-based WM data selection, noise-tolerant WM loss, and reward-adaptive loss balancing. Experiments on three agentic task benchmarks show consistent improvements over strong RL baselines across models and RL algorithms. These results suggest that standard RL rollouts are a practical source of WM supervision for language-agent training. Our code is available at https://github.com/ColinLu50/Policy-WorldModel-Agent.

1 Introduction

Reinforcement learning (RL) has become a dominant paradigm for improving large language model (LLM) agent performance (DeepSeek-AI, 2026; GLM-5-Team et al., 2026). However, standard RL optimizes actions for reward maximization without learning their consequences, leaving agents brittle to invalid operations, irreversible state changes, and delayed failures in long-horizon tasks (Hao et al., 2023; Liu et al., 2026). World modeling (WM) addresses this gap by predicting the next observation from the interaction history and the chosen action, encouraging the agent to internalize environment dynamics rather than memorize which actions get rewards (Zhang et al., 2025).

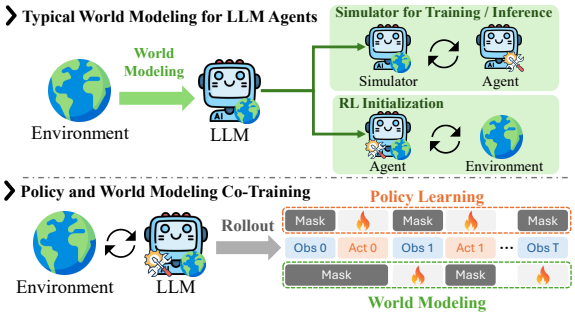


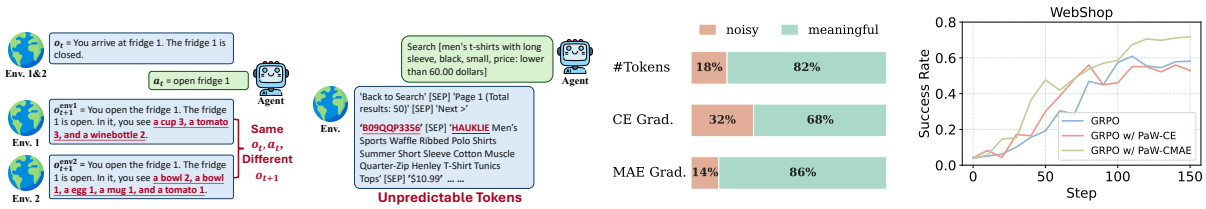
Figure 1: Comparison of world modeling paradigms for LLM agents. While prior methods rely on separate simulators, additional training, or inference-time planning, our PaW jointly optimizes policy learning and world modeling within the same model.

Existing WM methods for LLM agents typically introduce this ability outside the standard RL training, as shown in Figure 1. One line of work trains a world model simulator, either as a separate model or within the policy model itself, to generate imagined trajectories for RL training or to scale inference-time planning (Gu et al., 2025; Fang et al., 2025; Xiao et al., 2026; Li et al., 2025c; Liu et al., 2026). Another line of work first instills WM ability into the model and then fine-tunes it with RL training (Zhang et al., 2025; Yu et al., 2026). In both cases, WM learning incurs extra cost: a separate model, an additional training stage, or inference-time computation. This raises a question: *can world-modeling ability be learned jointly with policy improvement within the same RL training process?*

Our key observation is that on-policy RL rollouts already provide world-modeling supervision. Each interaction step yields both policy supervision from the action and its advantage, and dynamics supervision from the resulting next observation, which

*Equal contribution.
†Corresponding author: liusc3@sustech.edu.cn.

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 42962–42979
October 24-29, 2026 ©2026 Association for Computational Linguistics



(a) (b) (c) (d)

Figure 2: **Illustration of noisy observation tokens and the effect of clipped MAE loss.** (a) and (b) show two noisy WM training examples from ALFWorld and WebShop, where the same $(\boldsymbol{o}_t, \boldsymbol{a}_t)$ pair can lead to different next observation $\boldsymbol{o}_{t+1}$ in (a) and observations may contain random surface noise in (b). (c) shows that CE WM loss (Equation (5)) assigns a disproportionately large gradient share to noisy tokens. See Section A.4 for details. (d) shows that replacing CE (Equation (5)) with clipped MAE (Equation (9)) during training improves RL performance.

reveals what the action causes. While standard RL uses only the former, we exploits the latter as dense action-conditioned next-observation supervision, without requiring additional rollouts.

Motivated by this observation, we propose **PaW**, a framework for **P**olicy **a**nd **W**orld modeling co-training during on-policy RL post-training. As shown in Figures 1 and 3, PaW reuses RL rollouts by appending next-observation tokens and applying an auxiliary next-token-prediction loss to train the same model. Policy learning is unchanged because causal attention prevents later next-observation tokens from affecting the action logits. In inference, the agent behaves like a standard policy model, with no additional simulation steps.

However, as illustrated in Figures 2a and 2b, rollout observations provide noisy supervision: some transitions are uninformative, some target tokens are unpredictable, and the auxiliary WM loss must balance with the RL loss. Thus, PaW combines three key designs: action-entropy-based WM data selection, clipped mean absolute error (MAE;Ghosh et al. (2017)) loss for noisy observations, and reward-adaptive loss balancing. Together, they make auxiliary WM supervision both informative and stable. We evaluate PaW on two types of agentic tasks: interactive decision-making (ALFWorld and WebShop) and search-augmented QA. Across models and RL algorithms, PaW consistently improves strong RL baselines, with negligible training overhead. These results suggest that standard RL rollouts already provide a practical source of world-model supervision for language-agent training.

Our contributions are summarized as follows: (i) We identify next observations in on-policy rollouts as an overlooked source of action-conditioned WM supervision for language-agent RL; (ii) We propose **PaW**, the first policy and world-modeling co-training method for RL. It reuses on-policy rollouts for joint policy optimization and world-modeling supervision, with high-action-entropy transition selection, clipped MAE loss, and adaptive WM loss balancing; (iii) We show consistent improvements over strong RL baselines on three agentic tasks across models and RL algorithms.

2 Preliminaries

Problem setup. We consider language-agent tasks where a policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ solves a user-specified goal through multi-turn interaction with an environment. At turn t, the agent observes $\boldsymbol{o}_t$, forms a decision context $\boldsymbol{h}_t$ from the instruction and interaction history, and samples a textual or serialized action $\boldsymbol{a}_t \sim \boldsymbol{\pi}_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{h}_t)$. The environment executes $\boldsymbol{a}_t$ and returns reward r_t and next observation $\boldsymbol{o}_{t+1}$, yielding a trajectory $\boldsymbol{\tau} = \{\boldsymbol{o}_0, \boldsymbol{a}_0, r_0, \boldsymbol{o}_1, \ldots, \boldsymbol{o}_{T-1}, \boldsymbol{a}_{T-1}, r_{T-1}, \boldsymbol{o}_T\}$ with return $R(\boldsymbol{\tau}) = \sum_{t=0}^{T-1} r_t$.

On-policy Agentic RL. Agentic RL fine-tunes $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ to maximize $J(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\tau}\sim\boldsymbol{\pi}_{\boldsymbol{\theta}}}[R(\boldsymbol{\tau})]$ using sampled rollouts. Most on-policy algorithms can be abstracted as minimizing an advantage-weighted action loss:

$$\mathcal{L}_{\mathrm{RL}}(\boldsymbol{\theta}) = -\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=0}^{T-1} A_t \log \boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}_t \mid \boldsymbol{h}_t)\right], \quad (1)$$

where A_t is a reward-derived advantage and $\log \boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}_t \mid \boldsymbol{h}_t)$ sums token-level log-likelihoods of the action sequence. Different algorithms, such as GRPO (Shao et al., 2024) and GIGPO (Feng et al., 2025), mainly differ in how they estimate A_t and instantiate the surrogate objective. Our method is orthogonal to these choices and augments this base RL loss with world modeling supervision.

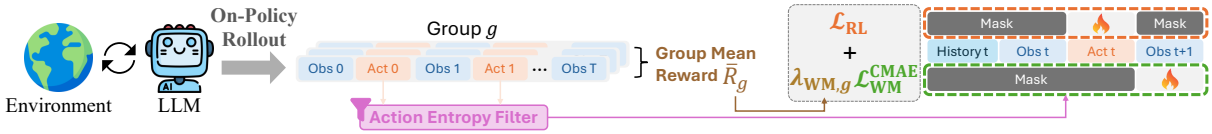


Figure 3: **Overview of PaW.** PaW introduces auxiliary world modeling to agentic RL via action-entropy WM data selection, clipped MAE, and reward-adaptive loss balancing.

World-modeling for language agents. World modeling aims to capture action-conditioned dynamics by predicting the environment response after an action (Zhang et al., 2025). For language agents, this becomes next-observation prediction: given $(\boldsymbol{h}_t, \boldsymbol{a}_t)$, an autoregressive model predicts the textual observation $\boldsymbol{o}_{t+1}$ with objective:

$$\mathcal{L}_{\text{WM}}(\boldsymbol{\phi}) = -\mathbb{E}\left[\log \boldsymbol{\pi}_{\boldsymbol{\phi}}(\boldsymbol{o}_{t+1} \mid \boldsymbol{h}_t, \boldsymbol{a}_t)\right], \quad (2)$$

where the likelihood is computed over observation tokens. Learning world modeling enables agents to better understand action outcomes and make better decisions in long-horizon tasks (Zhang et al., 2025; Liu et al., 2026).

3 Methodology

In this section, we present **PaW**, a framework for co-training policy learning with world modeling within a single policy model during on-policy RL by augmenting the base RL objective with an auxiliary WM objective:

$$\mathcal{L}_{\text{PaW}}(\boldsymbol{\theta}) = \mathcal{L}_{\text{RL}}(\boldsymbol{\theta}) + \lambda_{\text{WM}}\mathcal{L}_{\text{WM}}(\boldsymbol{\theta}). \quad (3)$$

Here, both terms update the same parameters $\boldsymbol{\theta}$: the RL loss improves action selection, while the WM loss trains next-observation prediction from rollout transitions. See Figure 3 for the overview. The rest of this section instantiates this objective by describing how we construct WM supervision with action-entropy-based data selection in Section 3.1, make observation prediction robust with clipped MAE in Section 3.2, set λ_{WM} as a reward-adaptive coefficient in Section 3.3, and summarize training and inference in Section 3.4.

3.1 Constructing World Modeling Supervision from RL Rollouts

At each RL update, we sample task instances and collect rollout groups $\mathcal{G}$ with the current policy, where each group $g \in \mathcal{G}$ contains trajectories from the same task. Each group forms a transition pool $\mathcal{P}_g = \{(\boldsymbol{h}_t, \boldsymbol{a}_t, r_t, \boldsymbol{o}_{t+1}) \mid \boldsymbol{\tau} \in g,\ 0 \le t < T\}$, and the update-level pool is $\mathcal{P} = \bigcup_{g\in\mathcal{G}} \mathcal{P}_g$.

Each transition in this pool serves two roles: $(\boldsymbol{h}_t, \boldsymbol{a}_t, A_t)$, with A_t computed from rewards, provides the action-level signal for policy learning, while $(\boldsymbol{h}_t, \boldsymbol{a}_t, \boldsymbol{o}_{t+1})$ provides the token-level target for next-observation prediction. However, using all transitions for WM learning can overweight redundant or low-informative action consequences. Thus, we select a subset of transitions for the auxiliary next-observation loss.

WM data selection. For each transition, let $q_{t,i} = \boldsymbol{\pi}_{\boldsymbol{\theta}}\left(\boldsymbol{a}_t^{(i)} \mid \boldsymbol{h}_t, \boldsymbol{a}_t^{(<i)}\right)$ denote the next-token probability at the i-th action position. We compute a sample-based estimate of the action-token entropy using the realized action tokens: $H(\boldsymbol{a}_t \mid \boldsymbol{h}_t) = -\frac{1}{|\boldsymbol{a}_t|}\sum_{i=1}^{|\boldsymbol{a}_t|} \log q_{t,i}$, a Monte Carlo estimate over the sampled action sequence. Intuitively, high-entropy actions correspond to decision points where the policy assigns probability mass to more diverse action alternatives. Their resulting observations are therefore more informative for learning action-conditioned environment transitions than highly deterministic, repetitive actions. We apply the selector to all candidate WM transitions in the current RL update. Given a retained fraction $\alpha \in (0, 1]$, we keep the top-α fraction by action entropy:

$$\mathcal{S}_\alpha = \text{Top}_\alpha\left(\mathcal{P};\ H(\boldsymbol{a}_t \mid \boldsymbol{h}_t)\right), \quad (4)$$

where Top_α returns the highest-entropy α fraction of $\mathcal{P}$. The RL loss is computed on all generated actions in the rollout groups, while the world-model loss is computed only on selected transitions in $\mathcal{S}_\alpha$.

Co-training in one forward pass. Operationally, each transition is serialized as $(\boldsymbol{h}_t, \boldsymbol{a}_t, \boldsymbol{o}_{t+1})$, with rewards used only for advantage computation. We apply an action-token mask for the base RL loss and an observation-token mask for the auxiliary loss on appended next-observation tokens in $\mathcal{S}_\alpha$. Causal attention prevents action tokens from attending to appended observations, so $\boldsymbol{o}_{t+1}$ does not affect action logits; meanwhile, observation logits are conditioned on $(\boldsymbol{h}_t, \boldsymbol{a}_t)$ for next-observation

prediction. The entropy uses action-token distributions already available in the rollout or training pass, introducing no additional model forward.

3.2 Clipped MAE Loss for Noisy Observation Prediction

After action-entropy-based selection, the selected transitions provide action-conditioned targets for world modeling. A direct instantiation is to apply the cross-entropy objective from Equation (2) to the next-observation tokens. For a selected transition $(\boldsymbol{h}_t, \boldsymbol{a}_t, r_t, \boldsymbol{o}_{t+1}) \in \mathcal{S}_\alpha$, let $p_{t,i} = \boldsymbol{\pi}_{\boldsymbol{\theta}}\left(\boldsymbol{o}_{t+1}^{(i)} \mid \boldsymbol{h}_t, \boldsymbol{a}_t, \boldsymbol{o}_{t+1}^{(<i)}\right)$ be the probability assigned to the i-th target token in $\boldsymbol{o}_{t+1}$. The standard CE loss on the selected set can be written as:

$$\mathcal{L}_{\mathrm{WM}}^{\mathrm{CE}}(\boldsymbol{\theta}; \mathcal{S}_\alpha) = \mathbb{E}_{\mathcal{S}_\alpha}\left[-\frac{1}{|\boldsymbol{o}_{t+1}|}\sum_{i=1}^{|\boldsymbol{o}_{t+1}|} \log p_{t,i}\right]. \tag{5}$$

Although CE is standard for language modeling, it is poorly matched to observation prediction in agentic environments. As illustrated in Figures 2a and 2b, next observations may be stochastic, non-unique, or contaminated by nuisance tokens such as IDs, product names, and random strings. Overfitting these tokens can consume optimization capacity without improving action-relevant dynamics.

MAE-style token loss. To reduce the influence of hard and low-probability observation tokens, we replace the per-token CE loss $\ell_{\mathrm{CE}}^{(i)} = -\log p_{t,i}$ with a mean MAE loss $\ell_{\mathrm{MAE}}^{(i)} = 1 - p_{t,i}$ (Ghosh et al., 2017). Let $z_{t,i}$ denote the logit assigned to the target token. The magnitudes of the corresponding target-logit gradients are:

$$\left|\frac{\partial \ell_{\mathrm{CE}}^{(i)}}{\partial z_{t,i}}\right| = 1 - p_{t,i}, \qquad \left|\frac{\partial \ell_{\mathrm{MAE}}^{(i)}}{\partial z_{t,i}}\right| = p_{t,i}(1 - p_{t,i}). \tag{6}$$

For a low-probability target token, the CE magnitude approaches one, whereas the MAE magnitude decreases proportionally with $p_{t,i}$. This makes MAE less sensitive to unpredictable observation fragments, as also reflected by the gradient-share analysis in Figure 2c. This loss enables WM learning, as shown in Section B.4.

Confidence clipping. MAE alone still keeps optimizing tokens after the model already predicts them with high confidence. However, in stochastic textual environments, forcing the model to further fit one observed realization may encourage memorization of arbitrary surface details. We therefore introduce a token-level confidence mask:

$$m_{t,i} = \mathbb{1}[p_{t,i} \le \rho], \tag{7}$$

where ρ is a confidence threshold. Tokens with $p_{t,i} > \rho$ are treated as sufficiently learned and are removed from the auxiliary target. For one selected transition, the clipped MAE loss is:

$$\ell_{\mathrm{CMAE}}(\boldsymbol{h}_t, \boldsymbol{a}_t, \boldsymbol{o}_{t+1}) = \frac{\sum_{i=1}^{|\boldsymbol{o}_{t+1}|} m_{t,i}(1 - p_{t,i})}{|\boldsymbol{o}_{t+1}|}. \tag{8}$$

The WM loss on the selected transitions is then:

$$\mathcal{L}_{\mathrm{WM}}^{\mathrm{CMAE}}(\boldsymbol{\theta}; \mathcal{S}_\alpha) = \mathbb{E}_{\mathcal{S}_\alpha}\left[\ell_{\mathrm{CMAE}}(\boldsymbol{h}_t, \boldsymbol{a}_t, \boldsymbol{o}_{t+1})\right]. \tag{9}$$

This loss focuses learning on insufficiently predicted observation tokens, while avoiding the excessive CE pressure on noisy or non-action-relevant tokens. Examples are shown in Section B.3.

3.3 Reward-Adaptive Loss Balancing

Even after entropy filtering and token-level clipping, WM supervision remains dense because every selected observation can contribute many token-level gradients. If applied with a fixed large weight, this auxiliary objective may dominate the sparse reward-driven policy update. Moreover, the need for auxiliary dynamics learning is task-dependent: low-performing rollout groups can benefit more from additional next-observation supervision, while high-performing groups should focus more on refining the policy objective.

We therefore instantiate the schematic coefficient λ_{WM} in Equation (3) as a reward-adaptive coefficient for each rollout group $g \in \mathcal{G}$. We set:

$$\lambda_{\mathrm{WM},g} = \left[1 - \frac{\bar{R}_g}{R_{\max}}\right]_+, \tag{10}$$

where where $[\cdot]_+$ denotes the positive-part operator, $\bar{R}_g = |g|^{-1}\sum_{\boldsymbol{\tau}\in g} R(\boldsymbol{\tau})$ denotes its mean episode return and $R_{\max}$ denotes the maximum attainable episode return in the environment. Low-return groups receive stronger WM supervision, whereas as $\bar{R}_g$ approaches $R_{\max}$, $\lambda_{\mathrm{WM},g}$ decreases, allowing the base RL objective to dominate.

3.4 Training and Inference

Combining the three designs above, the co-training objective of PaW in Equation (3) becomes the following final objective, which preserves the base

Type	Method	ALFWorld							WebShop	
		Pick	Look	Clean	Heat	Cool	Pick2	Avg.	Score	Succ.
Closed-Source Model										
Prompting	GPT-4o	75.3	60.8	31.2	56.7	21.6	49.8	48.0	31.8	23.7
Prompting	Gemini-2.5-Pro	92.8	63.3	62.1	69.0	26.6	58.7	60.3	42.5	35.9
Qwen2.5-1.5B-Instruct										
Prompting	Qwen2.5	5.9	5.5	3.3	9.7	4.2	0.0	4.1	23.1	5.2
Prompting	ReAct	17.4	20.5	15.7	6.2	7.7	2.0	12.8	40.1	11.3
Prompting	Reflexion	35.3	22.2	21.7	13.6	19.4	3.7	21.8	55.8	21.9
RL Training	GRPO	86.5	46.3	79.0	70.2	69.1	47.8	70.0	75.6	60.6
RL Training	**GRPO w/ PaW**	**87.8**	**59.3**	**84.5**	**73.7**	**75.4**	**69.6**	**77.9**	**83.8**	**68.6**
RL Training	GIGPO	95.3	**84.3**	87.7	**92.6**	79.8	82.3	87.6	83.2	66.2
RL Training	**GIGPO w/ PaW**	**95.3**	83.3	**91.8**	89.5	**89.1**	**84.5**	**90.4**	**87.7**	**75.3**
Qwen2.5-7B-Instruct										
Prompting	Qwen2.5	33.4	21.6	19.3	6.9	2.8	3.2	14.8	26.4	7.8
Prompting	ReAct	48.5	35.4	34.3	13.2	18.2	17.6	31.2	46.2	19.5
Prompting	Reflexion	62.0	41.6	44.9	30.9	36.3	23.8	42.7	58.1	28.8
RL Training	GRPO	90.8	66.1	**89.3**	74.7	72.5	64.7	77.6	75.4	66.5
RL Training	**GRPO w/ PaW**	90.4	**80.7**	86.8	**82.9**	**76.5**	**67.3**	**80.6**	**84.5**	**70.5**
RL Training	GIGPO	97.7	82.7	**98.8**	83.7	89.3	79.2	90.8	85.0	73.8
RL Training	**GIGPO w/ PaW**	**98.2**	**85.6**	98.6	**84.5**	**91.5**	**84.3**	**91.8**	**87.6**	**76.7**

Table 1: Performance on ALFWorld and WebShop. For ALFWorld, we report the success rate (%) for each subtask and the overall average. For WebShop, we report the average score and success rate (%). Bold numbers indicate the better result between each vanilla RL baseline and its PaW-augmented variant. Full results with standard variance can be found in Section B.1.

Algorithm 1 On-policy RL training with PaW.

for each RL update **do**
  Sample tasks and collect rollout groups $\mathcal{G}$ using $\boldsymbol{\pi}_{\boldsymbol{\theta}}$;
  Form transition pools $\{\mathcal{P}_g\}_{g\in\mathcal{G}}$ and $\mathcal{P} = \bigcup_g \mathcal{P}_g$;
  Compute selected transitions $\mathcal{S}_\alpha$ using Equation (4);
  for each rollout group $g \in \mathcal{G}$ **do**
    Set $\mathcal{S}_{\alpha,g} := \mathcal{S}_\alpha \cap \mathcal{P}_g$;
    Compute $\mathcal{L}_{\mathrm{RL}}(\boldsymbol{\theta}; g)$ using Equation (1);
    Compute $\mathcal{L}_{\mathrm{WM}}^{\mathrm{CMAE}}(\boldsymbol{\theta}; \mathcal{S}_{\alpha,g})$ using Equation (9);
    Compute $\lambda_{\mathrm{WM},g}$ using Equation (10);
  end for
  Optimize Equation (11) to update $\boldsymbol{\theta}$;
end for

on-policy RL loss over each rollout group while adding a reward-adaptively weighted CMAE world-model loss on the entropy-selected transitions:

$$\mathcal{L}_{\mathrm{PaW}}(\boldsymbol{\theta}) = \mathbb{E}_{g\in\mathcal{G}}\Big[\mathcal{L}_{\mathrm{RL}}(\boldsymbol{\theta}; g) + \lambda_{\mathrm{WM},g}\mathcal{L}_{\mathrm{WM}}^{\mathrm{CMAE}}(\boldsymbol{\theta}; \mathcal{S}_{\alpha,g})\Big]. \tag{11}$$

where $\mathcal{S}_{\alpha,g} = \mathcal{S}_\alpha \cap \mathcal{P}_g$, $\mathcal{L}_{\mathrm{RL}}(\boldsymbol{\theta}; g)$ is the base on-policy RL loss from Equation (1), and the auxiliary loss is computed only on $\mathcal{S}_{\alpha,g}$. If a group contains no selected transition, its auxiliary term is omitted.

Algorithm 1 summarizes one on-policy RL update with PaW. The procedure follows the base RL algorithm for rollout collection and advantage computation, then co-trains the same model with globally selected observation supervision and group-specific auxiliary weighting. Therefore, PaW requires no additional environment interaction, no separate model, and no separate training stage. It also introduces no additional model forward: the action entropy, action-token loss, and observation-token loss are computed from distributions already available during rollout generation or the masked training pass.

After training, PaW introduces no inference-time change. The model receives the decision context $\boldsymbol{h}_t$ and generates the next action $\boldsymbol{a}_t$ exactly as a standard policy model. It does not rollout imagined observations, perform planning with a simulator, or call an additional model. Thus, all benefits of WM co-training are obtained during training, while deployment keeps the same interface and cost as the underlying RL-trained agent.

4 Experiments

In this section, we empirically evaluate PaW on two types of agentic tasks, including interactive decision-making (i.e., ALFWorld and WebShop) and search-augmented QA.

Type	Method	Single-Hop QA			Multi-Hop QA				Avg.
		NQ†	TriviaQA⋆	PopQA⋆	HotpotQA†	2Wiki⋆	MuSiQue⋆	Bamboogle⋆	
Qwen2.5-3B-Instruct									
RL Training	Search-R1	34.1	54.5	37.8	32.4	31.9	10.3	26.4	32.5
RL Training	ZeroSearch	41.4	57.4	44.8	27.4	30.0	9.8	11.1	31.7
RL Training	GRPO	44.9	60.9	46.1	37.9	39.5	13.7	64.1	43.9
RL Training	**GRPO w/ PaW**	**45.8**	**61.2**	**47.5**	**39.4**	**40.1**	**14.4**	**65.2**	**44.8**
RL Training	GIGPO	42.5	58.5	46.3	35.2	34.4	11.8	60.0	41.2
RL Training	**GIGPO w/ PaW**	**46.2**	**61.8**	**46.7**	**37.6**	**38.0**	**13.9**	**64.9**	**44.2**
Qwen2.5-7B-Instruct									
RL Training	Search-R1	39.3	61.0	39.7	37.0	40.1	14.6	36.8	38.4
RL Training	ZeroSearch	43.6	61.8	51.5	34.6	35.2	18.4	27.8	39.0
RL Training	GRPO	47.9	63.9	47.8	43.9	43.6	18.3	69.6	47.9
RL Training	**GRPO w/ PaW**	**48.9**	**64.9**	**48.5**	**44.9**	**45.1**	**18.9**	**70.1**	**48.8**
RL Training	GIGPO	46.1	64.4	46.0	40.2	41.2	16.4	68.9	45.8
RL Training	**GIGPO w/ PaW**	**46.5**	**66.0**	**47.2**	**42.2**	**42.8**	**18.6**	**69.5**	**47.5**

Table 2: Performance on search-augmented QA tasks. Agents are trained on NQ and HotpotQA. † and ⋆ indicate in-domain and out-of-domain evaluation datasets, respectively. Avg. denotes the average score across all seven benchmarks. Bold indicates the better result between each vanilla RL baseline and its PaW-augmented variant.

4.1 Experiment Setup

Benchmarks. We evaluate PaW on interactive decision-making and search-augmented QA tasks. For interactive decision-making, we use ALFWorld (Shridhar et al., 2021), an embodied text-based environment with 3,827 household tasks instances across six household task categories: Pick & Place (Pick), Examine in Light (Look), Clean & Place (Clean), Heat & Place (Heat), Cool & Place (Cool), and Pick Two & Place (Pick2), and WebShop (Yao et al., 2022), a web shopping environment with over 110K products and 12K user instructions. For search-augmented QA, we evaluate multi-turn tool use on single-hop benchmarks including NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2022), as well as multi-hop benchmarks including HotpotQA (Yang et al., 2018), 2Wiki (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2022).

Baselines. We use GRPO (Shao et al., 2024) and GIGPO (Feng et al., 2025) as the main RL baselines, and compare each vanilla algorithm with its PaW-augmented variant. For ALFWorld and WebShop, we also compare against closed-source LLM agents, including GPT-4o (Achiam et al., 2023) and Gemini-2.5-Pro (Team et al., 2023), and prompting-based agents, including ReAct (Yao et al., 2023) and Reflexion (Shinn et al., 2023). For search-augmented QA, we additionally include Search-R1 (Jin et al., 2025) and ZeroSearch (Sun et al., 2025) as representative RL-based search-agent baselines.

RL	Model	Vanilla	w/ PaW
PPO	Qwen2.5-1.5B-Instruct	59.1	**65.2**$_{+6.1}$
RLOO	Qwen2.5-1.5B-Instruct	56.7	**61.2**$_{+4.5}$
GRPO	Qwen3-1.7B	45.4	**54.2**$_{+8.8}$
GRPO	Llama3.2-3B-Instruct	4.0	**62.2**$_{+58.2}$
GRPO	Qwen2.5-14B-Instruct	74.7	**77.1**$_{+2.4}$

Table 3: WebShop success rate (%) across different RL algorithms and base models.

Implementation details. Following (Feng et al., 2025), we use Qwen2.5-1.5B/7B-Instruct (Yang et al., 2024a) for ALFWorld and WebShop, and Qwen2.5-3B/7B-Instruct for search-augmented QA. We set the rollout group size to 8 for ALFWorld and WebShop and 5 for search-augmented QA, where E5 (Wang et al., 2022) is used as the retriever with at most 4 interaction turns. For PaW, we set entropy selection ratio to $\alpha = 0.75$ and the clipping threshold to $\rho = 0.2$, while keeping all RL hyperparameters identical to the corresponding vanilla RL baseline. All results are averaged over three random seeds. More details are shown in Section A.

4.2 Experimental Results

Table 1 shows that PaW consistently improves both GRPO and GIGPO on ALFWorld and WebShop across model scales. On ALFWorld, PaW improves the overall success rate of GRPO by +7.9 and +3.0 for Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct, respectively. It also improves GIGPO by +2.8 and +1.0. On WebShop, PaW yields even larger success-rate gains, improving GRPO by +8.0 and +4.0 and GIGPO by +9.1 and +2.9

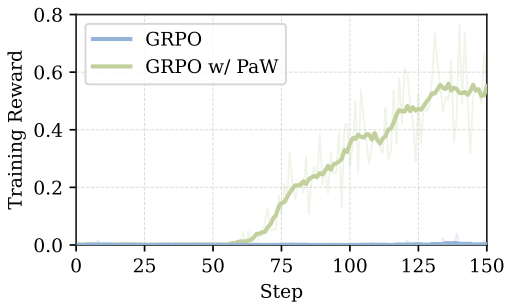


Figure 4: Training rewards of Llama3.2-3B-Instruct on WebShop. Compared with vanilla GRPO, PaW helps the agent escape sparse-reward failure and obtain positive success signals.

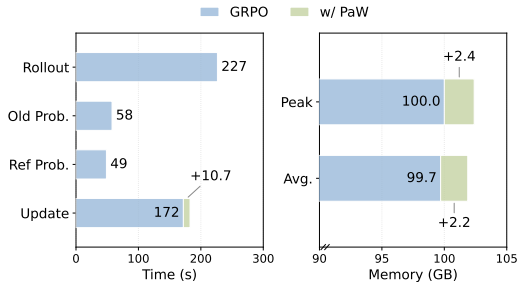


Figure 5: Per-step training time and GPU memory breakdown on ALFWorld with Qwen2.5-1.5B-Instruct. PaW increases both wall-clock time and GPU memory usage by approximately 2%.

at 1.5B and 7B, respectively. These results indicate that rollout-based world-model co-training improves long-horizon agent decision-making without adding extra models or changing inference.

Table 2 further evaluates PaW on multi-turn search-augmented QA tasks. For GRPO, PaW improves the average score from 43.9% to 44.8% at 3B and from 47.9% to 48.8% at 7B. For GIGPO, the average score increases from 41.2% to 44.2% at 3B and from 45.8% to 47.5% at 7B. The gains across interactive environments and search-augmented QA suggest that PaW is complementary to different RL algorithms and generalizes across agentic tasks. More results can be found in Section B.

4.3 Different Models and RL algorithms

We further examine whether the gains from PaW generalize beyond the main GRPO/GIGPO settings. To test generality across RL algorithms, we combine PaW with PPO (Schulman et al., 2017) and RLOO (Kool et al., 2019; Ahmadian et al., 2024). To test generality across model backbones, we apply GRPO w/ PaW to different model families and scales, including Qwen3-1.7B (Yang et al., 2025), Llama3.2-3B-Instruct (Grattafiori et al., 2024), and Qwen2.5-14B-Instruct. Implementation details can be found in Section A. As shown in Table 3, PaW consistently improves WebShop success rate across RL algorithms, model families, and model scales. It improves PPO and RLOO by +6.1 and +4.5, respectively, showing that the proposed world-model co-training objective is not tied to a specific RL algorithm. The improvement also persists across different backbones, including a +2.4 gain on Qwen2.5-14B-Instruct. Notably, PaW improves Llama3.2-3B-Instruct from 4.0% to 62.2%, suggesting that world-model supervision can provide useful learning signals when vanilla RL struggles to obtain positive rewards; we analyze this case in Section 4.4.

4.4 PaW Helps Where RL Fails

We next study whether PaW can help in sparse-reward settings where vanilla RL struggles to learn from on-policy rollouts. We train Llama3.2-3B-Instruct on WebShop with GRPO and compare the training dynamics with and without PaW. As shown in Figure 4, vanilla GRPO rarely obtains positive rewards in this challenging setting, causing the training signal to collapse. In contrast, PaW provides dense WM supervision by predicting next observations from collected state-action transitions. This auxiliary supervision allows the model to learn useful transition information even when most trajectories receive zero task reward. After several training steps, the model begins to generate successful rollouts, which in turn provides meaningful RL signals for further policy improvement. This result suggests that PaW can improve RL robustness by mitigating sparse-reward failures, especially for weaker base models or harder tasks. This analysis also corresponds to the +58.2 improvement on Llama3.2-3B-Instruct reported in Table 3.

4.5 Computational Overhead

We further measure the computational overhead introduced by PaW. We profile the per-step wall-clock time and GPU memory usage of vanilla GRPO and GRPO w/ PaW on ALFWorld with Qwen2.5-1.5B-Instruct. As shown in Figure 5, PaW introduces negligible overhead: it reuses the same rollout data and actor forward pass, adding only the WM objective during the RL update while leaving the rest of the pipeline unchanged. It adds only 10.7 s per step (2.1% of the $\sim$ 505 s GRPO

Method	ALFWorld	WebShop
GRPO	70.0	60.6
GRPO w/ PaW	**77.9**	**68.6**
w/o Ada. Coef.	75.5	67.0
w/ CE loss	68.5	57.2

Table 4: Ablation results on ALFWorld and WebShop with Qwen2.5-1.5B-Instruct using GRPO as the base RL algorithm. "w/o Ada. Coef." sets $\lambda_{\mathrm{WM},g} = 1$ for all rollout groups, and "w/ CE loss" replaces the CMAE observation loss with standard cross-entropy.

step time), with peak and average GPU memory increasing by 2.4 GB (2.4%) and 2.2 GB (2.2%), respectively. Thus, PaW improves agent performance with minimal training overhead. A detailed analysis is provided in Section A.5.

4.6 Ablation Study

Table 4 studies the reward-adaptive WM coefficient and the clipped MAE WM loss in PaW on ALFWorld and WebShop with Qwen2.5-1.5B-Instruct and GRPO . Fixing $\lambda_{\mathrm{WM},g} = 1$ still outperforms vanilla GRPO but reduces performance from 77.9% to 75.5% on ALFWorld and from 68.6% to 67.0% on WebShop, showing the benefit of adaptive loss balancing. Replacing CMAE with standard cross-entropy causes a larger drop, to 68.5% on ALFWorld and 57.2% on WebShop, indicating that CE can overfit noisy observation tokens while CMAE provides more robust supervision. Together, both components are important for effective world modeling co-training.

4.7 Valid Transitions Drive the Gains

We further examine whether the gains of PaW arise from learning valid action-conditioned transitions rather than from generic dense gradients. We compare two controls under the same Qwen2.5-1.5B-Instruct and GRPO setting: *AOShuffled* randomly permutes action–observation pairs, while *Random* replaces observation targets with random tokens. As shown in Table 5, both controls perform close to vanilla GRPO and substantially worse than PaW. The small improvement from AOShuffled may reflect auxiliary language-modeling regularization, while the full gain requires preserving the correct action–observation correspondence.

4.8 Generalization Performance

We further evaluate whether PaW generalizes beyond the environments observed during training. We train Qwen2.5-1.5B-Instruct on the

Method	ALFWorld	WebShop
GRPO	70.0	60.6
w/ PaW	**77.9**	**68.6**
w/ PaW (AOShuffled)	72.2	61.2
w/ PaW (Random)	69.8	60.8

Table 5: Success rates (%) under valid and corrupted transition supervision.

Method	ALFWorld-Unseen	WebShop	
	Succ.	Score	Succ.
GRPO	66.7	13.2	5.7
GRPO w/ PaW	**76.3**	**44.8**	**10.8**

Table 6: Generalization performance of Qwen2.5-1.5B-Instruct trained on the ALFWorld seen split.

ALFWorld seen split and evaluate it on unseen ALFWorld environments and on WebShop to assess cross-domain generalization. As shown in Table 6, PaW outperforms GRPO on both ALFWorld-Unseen (66.7% → 76.3%) and WebShop (5.7% → 10.8%), suggesting that the learned transition knowledge generalizes across environment instances and task domains.

4.9 Hyperparameter Sensitivity

We further analyze the sensitivity of PaW to the entropy selection ratio α and clipping threshold ρ on WebShop with Qwen2.5-1.5B-Instruct and GRPO. As shown in Figure 6, PaW remains effective across a broad range of values. Moderate clipping works best, with $\rho = 0.2$ performing the strongest, while overly large thresholds reduce performance, highlighting the importance of filtering unpredictable observation tokens. Performance also varies smoothly with α, with $\alpha = 0.75$ giving the best result, suggesting that entropy-based transition selection is useful but not overly sensitive to the exact ratio. Overall, PaW does not require delicate hyperparameter tuning.

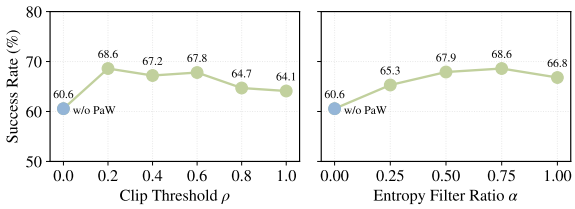


Figure 6: Hyperparameter sensitivity on WebShop with Qwen2.5-1.5B-Instruct using GRPO as the base RL algorithm. We vary the CMAE clipping threshold ρ and the entropy selection ratio α.

5 Related Work

Training LLM agents. LLM agents map instructions, interaction histories, and observations into executable actions for web, tool-use, embodied, and other interactive tasks (Wang et al., 2023; Yao et al., 2023; Shinn et al., 2023). Beyond prompting, recent work trains agents with supervised fine-tuning or reinforcement learning (Deng et al., 2023; Zeng et al., 2024; Chen et al., 2023; Xi et al., 2024; Jin et al., 2025; Sun et al., 2025; Feng et al., 2025). Because agent rewards are often sparse and delayed, existing RL methods mainly improve credit assignment or add auxiliary training signals (Feng et al., 2025; Fang et al., 2026; Lu et al., 2026). PaW is complementary to these approaches: instead of changing the policy-gradient estimator, it reuses the same on-policy rollouts to provide dense next-observation supervision.

World modeling for language agents. World models learn environment dynamics by predicting future states or rewards (Ha and Schmidhuber, 2018; Hafner et al., 2020, 2021). Recent language-agent methods use LLMs as world models, simulators, or transition predictors for planning, verification, and policy learning (Lin et al., 2024; Guo et al., 2025; Chae et al., 2025; Yu et al., 2026; Liu et al., 2026). While these methods demonstrate the value of future-observation modeling, they often require a separate world model, additional simulator training, or inference-time planning. In contrast, PaW folds world modeling into standard on-policy RL by training the same policy model to predict next observations from its own rollouts, introducing no extra interactions, no separate model, and no additional inference-time computation. Additional related work is discussed in Section C.

6 Conclusion

In this paper, we presented PaW, a policy and world modeling co-training framework for LLM agents. Rather than training a separate simulator or adding inference-time planning, PaW reuses on-policy RL rollouts as action-conditioned next-observation supervision and optimizes an auxiliary world-modeling loss on the same policy model. To make this supervision effective in noisy agentic environments, PaW combines action-entropy-based transition selection, clipped MAE observation prediction, and reward-adaptive loss balancing. Experiments on ALFWorld, WebShop, and search-augmented QA show consistent gains across RL algorithms and model scales, with minimal training overhead and no additional inference cost. These results suggest that standard RL rollouts provide a simple and practical source of world modeling supervision for improving language-agent training.

7 Limitations

While PaW consistently improves RL performance, there exist two main limitations. First, it relies on one-step next-observation supervision, which captures local dynamics but does not explicitly model longer-horizon dependencies or compounding prediction errors. Extending the co-training objective to multi-step world modeling is a promising direction for future work. Second, our experiments focus on relatively simple agent tasks such as web browsing. More complex scenarios, such as software engineering tasks, are left for future exploration.

Acknowledgments

This work was supported by National Key Research and Development Program of China under Grant 2022YFA1004102, and in part by the National Natural Science Foundation of China under Grants T2495254, 62502192.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to basics: Revisiting reinforce style optimization for learning from human feedback in LLMs. In *Annual Meeting of the Association for Computational Linguistics*.

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

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. 2023. FireAct: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*.

DeepSeek-AI. 2026. Deepseek-v4: Towards highly efficient million-token context intelligence.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2Web: Towards a generalist agent for the web. In *Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

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

Yangyi Fang, Jiaye Lin, Xiaoliang Fu, Cong Qin, Haolin Shi, Chang Liu, and Peilin Zhao. 2026. Proximity-based multi-turn optimization: Practical credit assignment for LLM agent training. *arXiv preprint arXiv:2602.19225*.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2025. Group-in-group policy optimization for LLM agent training. In *Conference on Neural Information Processing Systems*.

Aritra Ghosh, Himanshu Kumar, and P. S. Sastry. 2017. Robust loss functions under label noise for deep neural networks. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 1919–1925.

GLM-5-Team, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, and 167 others. 2026. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, Huan Sun, and Yu Su. 2025. Is your LLM secretly a world model of the internet? model-based planning for web agents. *Transactions on Machine Learning Research*.

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

David Ha and Jürgen Schmidhuber. 2018. World models. *arXiv preprint arXiv:1803.10122*.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. 2020. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. 2021. Mastering atari with discrete world models. In *International Conference on Learning Representations*.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *Conference on Empirical Methods in Natural Language Processing*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*.

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

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Conference on Language Modeling*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Wouter Kool, Herke van Hoof, and Max Welling. 2019. Buy 4 reinforce samples, get a baseline for free! In *International Conference on Learning Representations Workshop*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Hongxin Li, Jingfan Chen, Jingran Su, Yuntao Chen, Qing Li, and Zhaoxiang Zhang. 2025a. Autogui: Scaling gui grounding with automatic functionality annotations from llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pages 10323–10358.

Hongxin Li, Jingran Su, Jingfan Chen, Zheng Ju, Yuntao Chen, Qing Li, and Zhaoxiang Zhang. 2025b. Uipro: Unleashing superior interaction capability for gui agents. In *2025 IEEE/CVF International Conference on Computer Vision*, pages 1613–1623.

Yixia Li, Hongru Wang, Jiahao Qiu, Zhenfei Yin, Dongdong Zhang, Cheng Qian, Zeping Li, Pony Ma, Guanhua Chen, and Heng Ji. 2025c. From word to world: Can large language models be implicit text-based world models? *arXiv preprint arXiv:2512.18832*.

Jessy Lin, Yuqing Du, Olivia Watkins, Danijar Hafner, Pieter Abbeel, Dan Klein, and Anca Dragan. 2024. Learning to model the world with language. In *International Conference on Machine Learning*.

Youwei Liu, Jian Wang, Hanlin Wang, Beichen Guo, and Wenjie Li. 2026. Imagine-then-plan: Agent learning from adaptive lookahead with world models. *arXiv preprint arXiv:2601.08955*.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. 2026. Self-distilled agentic reinforcement learning. *arXiv preprint arXiv:2605.15155*.

Haoze Lv, Ning Lu, Ziang Zhou, Yew-Soon Ong, and Shengcai Liu. 2026. Ahd agent: Agentic reinforcement learning for automatic heuristic design. *arXiv preprint arXiv:2605.08756*.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, and 1 others. 2025. UI-TARS: Pioneering automated GUI interaction with native agents. *arXiv preprint arXiv:2501.12326*.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William E Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Kenji Toyama, Robert James Berry, Divya Tyamagundlu, Timothy P Lillicrap, and Oriana Riva. 2025. AndroidWorld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Learning Representations*.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Conference on Neural Information Processing Systems*.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*.

Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Fei Huang, and Yan Zhang. 2025. Zerosearch: Incentivize the search capability of LLMs without searching. *arXiv preprint arXiv:2505.04588*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. 2023. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Zhiyuan Wang, Shengcai Liu, Jiahao Wu, Ning Lu, Hui Ouyang, Shaofeng Zhang, Haoze Lv, and Ke Tang. 2026. Cooperative coevolution for resource-constrained agentic llm post-training. *arXiv preprint arXiv:2608.02391*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Conference on Neural Information Processing Systems*.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, and 1 others. 2024. AgentGym: Evolving large language model-based agents across diverse environments. In *Annual Meeting of the Association for Computational Linguistics*.

Zikai Xiao, Jianhong Tu, Chuhang Zou, Yuxin Zuo, Zhi Li, Peng Wang, Bowen Yu, Fei Huang, Junyang Lin,

and Zuozhu Liu. 2026. WebWorld: A large-scale world model for web agent training. *arXiv preprint arXiv:2602.14721*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024a. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. 2024b. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Conference on Neural Information Processing Systems*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Conference on Neural Information Processing Systems*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*.

Xiao Yu, Baolin Peng, Ruize Xu, Yelong Shen, Pengcheng He, Suman Nath, Nikhil Singh, Jiangfeng Gao, and Zhou Yu. 2026. Reinforcement world model learning for LLM-based agents. *arXiv preprint arXiv:2602.05842*.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2024. AgentTuning: Enabling generalized agent abilities for LLMs. In *Findings of the Annual Meeting of the Association for Computational Linguistics*.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, Jian Xie, Yuxuan Sun, Boyu Gou, Qi Qi, Zihang Meng, Jianwei Yang, Ning Zhang, Xian Li, Ashish Shah, and 11 others. 2025. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*.

A Implementation Details

A.1 Details of Training

Hyperparameters for ALFWorld. All methods are configured with identical hyperparameters: the maximum prompt length is 2048 tokens, and the maximum response length is 512 tokens. Each episode allows up to 50 environment steps. The learning rate is set to 1e-6 for the actor and 1e-5 for the critic (used only in PPO). We adopt a rule-based reward, assigning a reward of 10 for success and 0 for failure ($R_{\text{max}} = 10$). To handle invalid actions generated by the agent, we apply a reward penalty of -0.1. For all group-based RL methods, we use a group size of 8 and sample 16 different groups per rollout, resulting in a total of 128 environments. In contrast, PPO uses 128 separate environments for rollouts. For GIGPO, the weighting coefficient ω is fixed at 1 without further tuning, and the discount factor γ is set to 0.95. And we use the normalized version of GIGPO. The rollout temperature is set to 1.0, while the validation temperature is set to 0.4. The mini-batch size is 256 The history number is set to 2.

Hyperparameters for WebShop. All methods are configured with identical hyperparameters: the maximum prompt length is 4096 tokens, and the maximum response length is 512 tokens. Each episode is limited to 15 environment steps. The learning rate is 1e-6 for the actor and 1e-5 for the critic (used only in PPO). We adopt a rule-based reward, assigning a reward of 10 for success and 0 for failure. So the $R_{\text{max}} = 10$. Invalid actions are penalized with a reward of -0.1. As with ALFWorld, all group-based RL methods use a group size of 8 and sample 16 groups per rollout, totaling $16 \times 8 = 128$ environments. PPO, on the other hand, uses 128 distinct environments for rollouts. For GIGPO, the weighting coefficient ω is set to 1 without additional tuning, and the discount factor γ is set to 0.95. And we use the normalized version of GIGPO. The rollout temperature is set to 1.0, while the validation temperature is set to 0.4. The mini-batch size is 64.

Hyperparameters for Search-Augmented QA. The maximum prompt length is 4096 tokens, and the maximum response length is 512 tokens. The max turn is set to 4. The learning rate is 1e-6 for the actor. We adopt a rule-based reward, assigning a reward of 1 for success and 0 for failure. So the $R_{\text{max}} = 1$. Invalid actions are penalized with a reward of -0.01. We set the train data size to 256 and use a group size of 5. For GIGPO, the weighting coefficient ω is set to 1 without additional tuning, the discount factor γ is set to 0.95, and the similarity threshold is set to 0.9. Rollout and validation temperatures are set to 1.0 and 0.0, respectively. The mini-batch size is 512.

Prompt Template for ALFWorld

You are an expert agent operating in the ALFRED embodied Environment. Your task is to: {task_description}. Prior to this step, you have already taken {step_count} step(s). Below are the most recent {history_length} observations and the corresponding actions you took: {action_history}. You are now at step {current_step} and your current observation is: {current_observation}. Your admissible actions of the current situation are: [{admissible_actions}].
Now it's your turn to take an action. You should first reason step-by-step about the current situation. This reasoning process MUST be enclosed within <think> </think> tags. Once you've finished your reasoning, you should choose an admissible action for current step and present it within <action> </action> tags.

Figure 7: The prompt template of ALFWorld agents.

A.2 Baseline Details

- *GPT-4o:* A closed-source, large-scale LLM used as a baseline for multi-turn agentic tasks (Achiam et al., 2023).
- *Gemini-2.5-Pro:* Another closed-source LLM, comparable in scale and capability to GPT-4o (Team et al., 2023).
- *ReAct:* A prompting-based agent that integrates reasoning and acting in an interleaved chain-of-thought framework (Yao et al., 2023).
- *Reflexion:* A prompting agent that incorporates self-reflection and iterative improvement over generated outputs (Shinn et al., 2023).
- *PPO:* Proximal Policy Optimization, a classic RL algorithm for policy learning (Schulman et al., 2017).
- *RLOO:* Reinforcement Learning with Offline Observations, a group-based RL approach that estimates advantages without value networks (Kool et al., 2019; Ahmadian et al., 2024).

Prompt Template for WebShop

You are an expert autonomous agent operating in the WebShop e-commerce environment. Your task is to: {task_description}. Prior to this step, you have already taken {step_count} step(s). Below are the most recent {history_length} observations and the corresponding actions you took: {action_history}. You are now at step {current_step} and your current observation is: {current_observation}. Your admissible actions for the current situation are: [{available_actions}].
Now it's your turn to take one action for the current step. You should first reason step-by-step about the current situation, then think carefully which admissible action best advances the shopping goal. This reasoning process MUST be enclosed within <think> </think> tags. Once you've finished your reasoning, you should choose an admissible action for current step and present it within <action> </action> tags.

Figure 8: The prompt template of WebShop agents.

Prompt Template for Search

You are an expert agent tasked with answering the given question step-by-step. Your question: {task_description}. Prior to this step, you have already taken {step_count} step(s). Below is the interaction history where <search> </search> wrapped your past search queries and <information> </information> wrapped the corresponding search results returned by the external search engine. History: {memory_context}
Now it's your turn to respond for the current step. You should first conduct reasoning process. This process MUST be enclosed within <think> </think> tags. After completing your reasoning, choose only one of the following actions (do not perform both):
(1) If you find you lack some knowledge, you can call a search engine to get more external information using format: <search> your query </search>.
(2) If you have enough knowledge to answer the question confidently, provide your final answer within <answer> </answer> tags, without detailed illustrations. For example, <answer>Beijing</answer>.

Figure 9: The prompt template of Search agents.

- *GRPO:* Group-based RL with trajectory-level advantage estimation, designed to scale RL to multi-step tasks (Shao et al., 2024).
- *GiGPO:* Grouped Incremental GPO, a prior hierarchical RL method that performs group-wise advantage estimation for LLM-based agents (Feng et al., 2025).

A.3 Prompts

The prompt templates used for the LLM agents are shown in Figure 7, Figure 8, and Figure 9. Each template is implemented using Python-style string formatting, where fields enclosed in curly braces ({}) denote semantic slots that are instantiated at runtime via Python's `.format()` function. For example, placeholders such as {task_description}, {step_count}, and {current_observation} are dynamically replaced with task-specific context. To provide the agent with temporal context, we additionally incorporate interaction history: we retain the two most recent history steps for ALFWorld and WebShop, and use the complete history for search-augmented question answering.

The <think> </think> block is used to elicit explicit step-by-step reasoning from the agent, encouraging chain-of-thought (Wei et al., 2022) style deliberation. The <action> </action> block specifies the agent's final action decision. For the search agent, reasoning traces are produced within <think> </think> tags, search queries are issued within <search> </search> tags, and final answers are provided within <answer> </answer> tags. Retrieved evidence from the retriever is supplied to the agent using <information> </information> tags.

A.4 Noise-Gradient Analysis

This section describes the diagnostic used for panel (c) of Figure 2. The goal is to measure how much of the world-model gradient budget is assigned to unpredictable WebShop observation tokens under CE and MAE losses. We run the diagnostic on WebShop with Qwen2.5-1.5B-Instruct. We randomly sample 100 WebShop search-result steps from rollout traces. For each sampled transition $(\boldsymbol{h}_t, \boldsymbol{a}_t, \boldsymbol{o}_{t+1})$, we use the same serialization format as world-model training. The decision context $\boldsymbol{h}_t$ and the generated action $\boldsymbol{a}_t$ form the prefix, and

Type	Method	ALFWorld							WebShop	
		Pick	Look	Clean	Heat	Cool	Pick2	All	Score	Succ.
Qwen2.5-1.5B-Instruct										
RL Training	GRPO	$86.5_{\pm 4.3}$	$46.3_{\pm 8.4}$	$79.0_{\pm 3.2}$	$70.2_{\pm 2.5}$	$69.1_{\pm 8.0}$	$47.8_{\pm 5.8}$	$70.0_{\pm 4.2}$	$75.6_{\pm 3.8}$	$60.6_{\pm 4.6}$
RL Training	**GRPO w/ PaW**	$\mathbf{87.8}_{\pm 3.6}$	$\mathbf{59.3}_{\pm 8.9}$	$\mathbf{84.5}_{\pm 1.7}$	$\mathbf{73.7}_{\pm 4.3}$	$\mathbf{75.4}_{\pm 8.4}$	$\mathbf{69.6}_{\pm 3.5}$	$\mathbf{77.9}_{\pm 2.7}$	$\mathbf{83.8}_{\pm 2.1}$	$\mathbf{68.6}_{\pm 2.9}$
RL Training	GIGPO	$95.3_{\pm 2.2}$	$84.3_{\pm 4.6}$	$87.7_{\pm 1.2}$	$92.6_{\pm 7.0}$	$79.8_{\pm 1.4}$	$82.3_{\pm 6.1}$	$87.6_{\pm 1.6}$	$83.2_{\pm 1.9}$	$66.2_{\pm 3.4}$
RL Training	**GIGPO w/ PaW**	$\mathbf{95.3}_{\pm 4.5}$	$83.3_{\pm 4.5}$	$\mathbf{91.8}_{\pm 6.0}$	$89.5_{\pm 4.3}$	$\mathbf{89.1}_{\pm 0.3}$	$\mathbf{84.5}_{\pm 5.8}$	$\mathbf{90.4}_{\pm 1.3}$	$\mathbf{87.7}_{\pm 1.7}$	$\mathbf{75.3}_{\pm 3.2}$
Qwen2.5-7B-Instruct										
RL Training	GRPO	$90.8_{\pm 5.1}$	$66.1_{\pm 6.7}$	$89.3_{\pm 5.4}$	$74.7_{\pm 6.9}$	$72.5_{\pm 5.4}$	$64.7_{\pm 7.3}$	$77.6_{\pm 5.2}$	$75.4_{\pm 2.7}$	$66.5_{\pm 3.6}$
RL Training	**GRPO w/ PaW**	$90.4_{\pm 6.8}$	$\mathbf{80.7}_{\pm 7.4}$	$86.8_{\pm 6.8}$	$\mathbf{82.9}_{\pm 8.6}$	$\mathbf{76.5}_{\pm 2.6}$	$\mathbf{67.3}_{\pm 3.7}$	$\mathbf{80.6}_{\pm 3.1}$	$\mathbf{84.5}_{\pm 2.3}$	$\mathbf{70.5}_{\pm 2.9}$
RL Training	GIGPO	$97.7_{\pm 1.6}$	$82.7_{\pm 7.9}$	$98.8_{\pm 1.6}$	$83.7_{\pm 7.2}$	$89.3_{\pm 8.2}$	$79.2_{\pm 6.6}$	$90.8_{\pm 1.3}$	$85.0_{\pm 2.9}$	$73.8_{\pm 3.2}$
RL Training	**GIGPO w/ PaW**	$\mathbf{98.2}_{\pm 1.0}$	$\mathbf{85.6}_{\pm 6.2}$	$98.6_{\pm 1.1}$	$\mathbf{84.5}_{\pm 7.6}$	$\mathbf{91.5}_{\pm 7.2}$	$\mathbf{84.3}_{\pm 6.8}$	$\mathbf{91.8}_{\pm 1.2}$	$\mathbf{87.6}_{\pm 2.4}$	$\mathbf{76.7}_{\pm 2.7}$

Table 7: Full performance on ALFWorld and WebShop with statistical stats. Results are averaged over 3 random seeds. For ALFWorld, we report the average success rate (%) for each subtask as well as the overall result. For WebShop, we report both the average score and the average success rate (%).

the next observation $\boldsymbol{o}_{t+1}$ is treated as the teacher-forced target. We only score tokens that belong to the next-observation span.

For the i-th target token in $\boldsymbol{o}_{t+1}$, we compute the model probability $p_{t,i} := \boldsymbol{\pi_\theta}\left(\boldsymbol{o}_{t+1}^{(i)} \mid \boldsymbol{h}_t, \boldsymbol{a}_t, \boldsymbol{o}_{t+1}^{(<i)}\right)$, following the notation of Section 3.2. We then compare two token-level losses. The CE diagnostic uses $\ell_{\text{CE}}^{(i)} = -\log p_{t,i}$, matching the unbounded WM CE objective in Equation (5). The MAE diagnostic uses $\ell_{\text{MAE}}^{(i)} = 1 - p_{t,i}$, matching the bounded per-token term used by our Clipped MAE loss in Equation (8) before applying the confidence mask $m_{t,i}$. To compare gradient allocation rather than raw loss scale, we compute the target-logit gradient magnitude for each token, consistent with the parameter-gradient comparison in Equation (6). For CE this magnitude is $1 - p_{t,i}$. For MAE this magnitude is $p_{t,i}(1 - p_{t,i})$. We sum these magnitudes within each token category and normalize them by the total gradient magnitude over all scored next-observation tokens.

We define noisy tokens using human-written WebShop rules. A token is labeled as noisy if its character span overlaps either a random product identifier or a brand string. Random product identifiers are matched with ASIN-style pattern `B[0-9A-Z]{9}`. Brand strings are matched as brand-like uppercase spans with at least four letters, optionally followed by digits. This threshold excludes structural markers such as `SEP`. All other next-observation tokens are labeled as meaningful for this diagnostic. When a word is split into multiple subword tokens, each subword inherits noisy label if its offset overlaps a noisy character span.

This analysis is evaluation-only and does not update model parameters. It isolates the effect of the loss function on gradient allocation over the same sampled tokens. The resulting normalized shares show whether a loss over-allocates gradient to noisy WebShop surface strings.

A.5 Computational Overhead of PaW

Overall overhead. PaW introduces no additional trainable parameters and increases training time and memory usage by only approximately 2%, as reported in Figure 5.

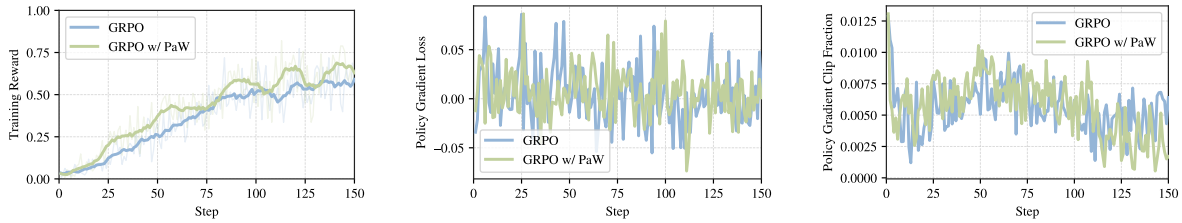
Stage-wise analysis. We decompose the training pipeline to identify where PaW incurs additional computation relative to vanilla on-policy RL.

- **Rollout (no overhead).** Both methods use the same standard generation procedure.
- **Old log-probability computation (no overhead).** Both methods perform the same computation required by the base RL objective.
- **Policy update (additional overhead).** Vanilla RL computes the loss over action tokens, whereas PaW additionally processes observation tokens and backpropagates the world-modeling loss. This is the only source of additional computation.
- **KL regularization (no overhead).** Both methods use the same current- and reference-policy computations.

B Additional Experimental Results

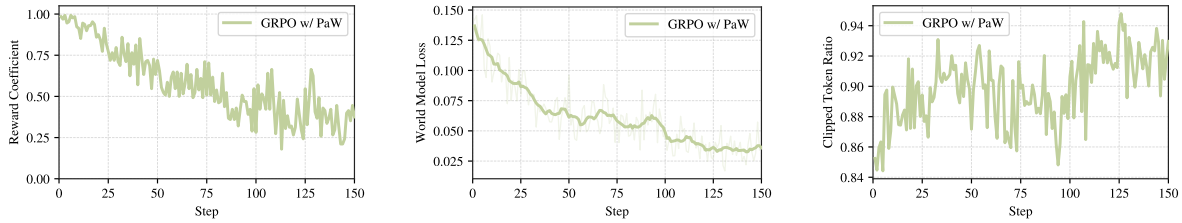
B.1 Full Results of ALFWorld and WebShop

Table 7 shows the full results on ALFWorld and WebShop, including the statistical stats. Results



(a) Training reward. (b) Policy-gradient loss. (c) Clipped update ratio.

Figure 10: Policy-side training dynamics on WebShop. PaW improves the training reward over the GRPO baseline, while the policy-gradient loss and clipped update ratio remain broadly comparable. This suggests that the auxiliary world modeling objective improves learning without substantially changing the main policy-optimization dynamics.



(a) Adaptive coefficient. (b) World modeling loss. (c) Clipped token ratio.

Figure 11: PaW-side training dynamics. The reward-adaptive coefficient decreases as training reward improves, the world modeling loss trends downward, and the clipped token ratio reports how often the margin objective clips observation-token supervision.

are averaged over 3 random seeds. It shows that PaW consistently improves the performance over the base RL methods.

B.2 Training Dynamics

Figure 10 compares the policy-side dynamics. GRPO w/ PaW obtains better training reward than the GRPO baseline, indicating that the additional world-model supervision improves the learned policy during RL. At the same time, the policy-gradient loss and clipped update ratio are close to those of GRPO, suggesting that the improvement does not come from a large change in PPO-style update magnitude or clipping behavior.

Figure 11 further analyzes the auxiliary world-model objective. The adaptive coefficient decreases during training, because higher-reward rollouts receive less world-model supervision under the reward-adaptive scaling rule. Meanwhile, the world-model loss decreases, showing that the shared policy model learns the next-observation prediction target from RL rollouts. The clipped token ratio tracks how many observation tokens are affected by the margin clipping mechanism, which keeps the auxiliary signal bounded as the world-model prediction improves.

B.3 Analysis of Observation Tokens

We analyze which observation tokens receive reduced supervision under the clipped MAE objective. Using the step-150 checkpoint of Qwen2.5-1.5B-Instruct, we collect token-level WM probabilities from 100 WebShop transitions, containing approximately 37K observation tokens. Among 1,000 sampled instances with the lowest target-token probabilities, 81% correspond to noisy surface details, primarily product identifiers, while only 0.2% contain task-critical information according to human evaluation. Among 500 token instances immediately above the clipping threshold, 79% are identifiers, numbers, or prices. These statistics indicate that clipped MAE primarily reduces optimization pressure on unpredictable surface details rather than decision-relevant state information. Representative examples are shown in Table 8.

B.4 PaW Learns WM Knowledge

We further evaluate whether the model acquires action-conditioned transition knowledge under clipped MAE in PaW. Given an interaction history and the current action, we evaluate perplexity on the ground-truth next-observation tokens (PPL; lower is better) and accuracy on an eight-way multiple-choice prediction task (MCP Acc.;

Category	Token	Context	Probability
Lowest probability	JR	[SEP] 'B07DKG**JR**74' [SEP] 'Amazon Brand - Daily Ritual'	9.4×10^{-13}
	CBD	[SEP] 'B00IT**CBD**3Y' [SEP] 'Laredo Womens Spellbound Studded'	7.0×10^{-12}
	HR	[SEP] 'B09Q5Z**HR**VM' [SEP] 'Yinimo Mens Gym Shorts Fashionable'	1.0×10^{-11}
Near threshold	2	Happy Birthday Mickey Confetti T-Shirt [SEP] '$2**2**.99'	0.2
	0	Fit V-Neck Pullover Sweater [SEP] '$2**0**.66 to $29.2' [SEP]	0.2
	2	Raglan Baseball Tee [SEP] '$**2**3.99' [SEP]	0.2

Table 8: Representative low-probability and near-threshold WebShop observation tokens. Square brackets mark the target token within its context.

Model	PPL ↓	MCP Acc. ↑
Base	1.81	0.28
GRPO	1.98	0.22
GRPO w/ PaW	**1.38**	**0.42**

Table 9: Next-observation prediction on 256 unseen ALFWorld cases.

higher is better). MCP distractors are generated by executing alternative actions from the same state. We evaluate Qwen2.5-1.5B-Instruct on 256 unseen ALFWorld cases during training.

As shown in Table 9, GRPO degrades both metrics relative to the base model, whereas GRPO w/ PaW reduces PPL to 1.38 and improves MCP accuracy to 0.42. This confirms that the model learns useful WM knowledge under the clipped MAE WM loss.

C Additional Related Work

LLM-based agents. Large language models have been widely used as autonomous agents that translate natural-language instructions, interaction histories, and environment observations into executable actions (Wang et al., 2023; Yao et al., 2023; Shinn et al., 2023). Such agents have been applied to a broad range of interactive domains, including embodied navigation and open-ended games (Wang et al., 2024), software engineering (Jimenez et al., 2024; Yang et al., 2024b), browser and web navigation (Zhou et al., 2024; Qin et al., 2025; Deng et al., 2023), algorithm design (Lv et al., 2026), and system control (Rawles et al., 2025; Li et al., 2025a,b). Early agent systems mainly rely on prompting, tool use, memory, planning, or reflection to elicit multi-step behavior from frozen LLMs (Yao et al., 2023; Shinn et al., 2023). Although these approaches can produce strong zero-shot or few-shot behaviors, their performance is often limited by distribution shift, long-horizon error accumulation, and the lack of task-specific adaptation.

Training language agents. To improve agent behavior beyond prompting, recent work trains LLM agents with supervised fine-tuning (SFT) or reinforcement learning (RL). SFT-based methods train agents on human demonstrations or trajectories synthesized by stronger LLMs, providing dense action-level supervision (Deng et al., 2023; Zeng et al., 2024; Chen et al., 2023; Xi et al., 2024). However, collecting high-quality demonstrations can be expensive, and synthetic trajectories remain constrained by the coverage, quality, and biases of the teacher model or hand-designed workflow. RL-based methods instead optimize agents using environment rewards or verifier feedback, allowing policies to improve through trial and error without step-by-step demonstrations (Jin et al., 2025; Sun et al., 2025; Wang et al., 2026; Feng et al., 2025). A central challenge in agentic RL is sparse and delayed rewards. Recent methods address this through improved credit assignment or auxiliary supervision. GIGPO, for example, estimates relative advantages at both the episode and step levels (Feng et al., 2025).

World modeling and language-agent learning. World models have a long history in reinforcement learning, where they are trained on observed transitions to predict future states and rewards (Ha and Schmidhuber, 2018; Hafner et al., 2020, 2021). By learning environment dynamics, world models can support planning, improve sample efficiency, and provide richer learning signals. Recent work extends world modeling to language agents, where states and observations are represented as text and transitions are induced by language actions. RAP frames reasoning with language models as planning with an internal world model (Hao et al., 2023). Dyna-style language-agent methods train action-conditioned transition models or align simulated next states with realized next states (Lin et al., 2024;

Guo et al., 2025; Yu et al., 2026). Other work uses language-based simulators to support web-agent planning, verification, or data generation (Gu et al., 2025; Chae et al., 2025). Imagine-then-Plan uses a learned world model to generate imagined trajectories and adaptively choose the planning horizon (Liu et al., 2026). These studies demonstrate that predicting future observations can improve agent reasoning and decision making.

D Licenses and Data Privacy

Artifact	Usage	License
verl-agent	RL training framework.	Apache License 2.0
ALFWorld	Embodied agent environment.	MIT License
WebShop	Web-shopping agent environment.	MIT License
Search-R1	Search/tool-use benchmark and data processing.	Apache License 2.0

Table 10: Licenses of external artifacts used in this work.

All models, datasets, and code artifacts used in this work are used in compliance with their respective licenses, and we retain the original attribution and license notices. We verified that the dataset used in this work does not contain personally identifying information or offensive content. The data are fake environments, with no names, contact information, addresses, user identifiers, or other unique personal identifiers. Therefore, the issue of anonymizing personal information is not applicable to this study.

E Use of LLMs

We used LLMs exclusively as writing assistants to refine language. In particular, their use was restricted to grammar correction, style improvement, and phrasing adjustments for clarity and conciseness.