FROM OUTCOMES TO STEPS: RELIABLE SUCCESS-AWARE CRITIC LEARNING FOR LONG-HORIZON LANGUAGE AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning for long-horizon language agents requires identifying whether intermediate actions make eventual task success more or less likely when supervision is available only at trajectory level. The main difficulty is that *a critic with aggregate return-prediction accuracy may still fail to resolve the local value differences needed to assign credit to actions*, especially under sparse feedback and function approximation. Conventional critics estimate future return, but policy improvement also depends on knowing whether each intermediate decision makes task success more or less likely. We introduce `SCALE`, an actor–critic framework that augments conventional return estimation with explicit prediction of eventual task success. At each interaction prefix, the critic predicts a return value and a **Success Value**, defined as the probability that the task will eventually be completed from that prefix. `SCALE` converts changes in Success Value into multi-step action credit through **Success-GAE** and modulates its contribution according to the estimated reliability of the predictor, while retaining the token-level PPO objective. Across five agent benchmarks and three model backbones, `SCALE` improves task performance over the compared RL baselines in most settings, with gains of up to **4.1** percentage points in success rate. Separate critic-learning analyses show that `SCALE` also improves critic estimation and training stability.

1 INTRODUCTION

Large language models (LLMs) are deployed as interactive agents that browse the web, use tools, and interact with environments over many steps (Yao et al., 2023; Shridhar et al., 2021; Yao et al., 2022). Reinforcement learning (RL) provides a framework for improving such agents from task outcomes (Jin et al., 2025; Feng et al., 2026; Wang et al., 2025b; 2026b), yet it remains difficult to tell which intermediate actions help or hurt the outcome. As illustrated in Fig. 1, an action can satisfy a local constraint yet make the task less likely to succeed. The learner therefore needs to determine which intermediate decisions improve the prospect of eventual task completion. For long-horizon language agents, this is difficult because supervision often arrives only after the entire interaction, while policy improvement depends on distinguishing the effects of individual decisions along the way.

Existing methods assign credit at different parts of the trajectory. Standard PPO and outcome-based methods optimize token probabilities from trajectory-level returns (Schulman et al., 2017; Ouyang et al., 2022; Shao et al., 2024), while recent methods assign credit to individual turns or environment actions (Zeng et al., 2025; Feng et al., 2025; Li et al., 2025b; Wang et al., 2026a; Ding et al., 2026). Critic-based methods further estimate intermediate values to propagate delayed outcomes (Zhou et al., 2024; Kazemnejad et al., 2025; Hou et al., 2026). These methods assign credit at finer parts of the trajectory, but accurate prediction of future return does not necessarily imply reliable action-level credit. Under sparse supervision and function approximation, a change in predicted return may not tell us whether a particular action made eventual success more likely. The central issue is therefore not only where credit is assigned, but *whether the critic preserves the local ordering between alternative actions* and *whether its reliability can be assessed during training*. In Sec. 3.2, we show that this concern is practical rather than hypothetical, conventional PPO critics produce action-level credit that degrades toward random chance at long horizons and remains unstable throughout training, leaving reliable critic learning for long-horizon credit assignment an open challenge.

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.

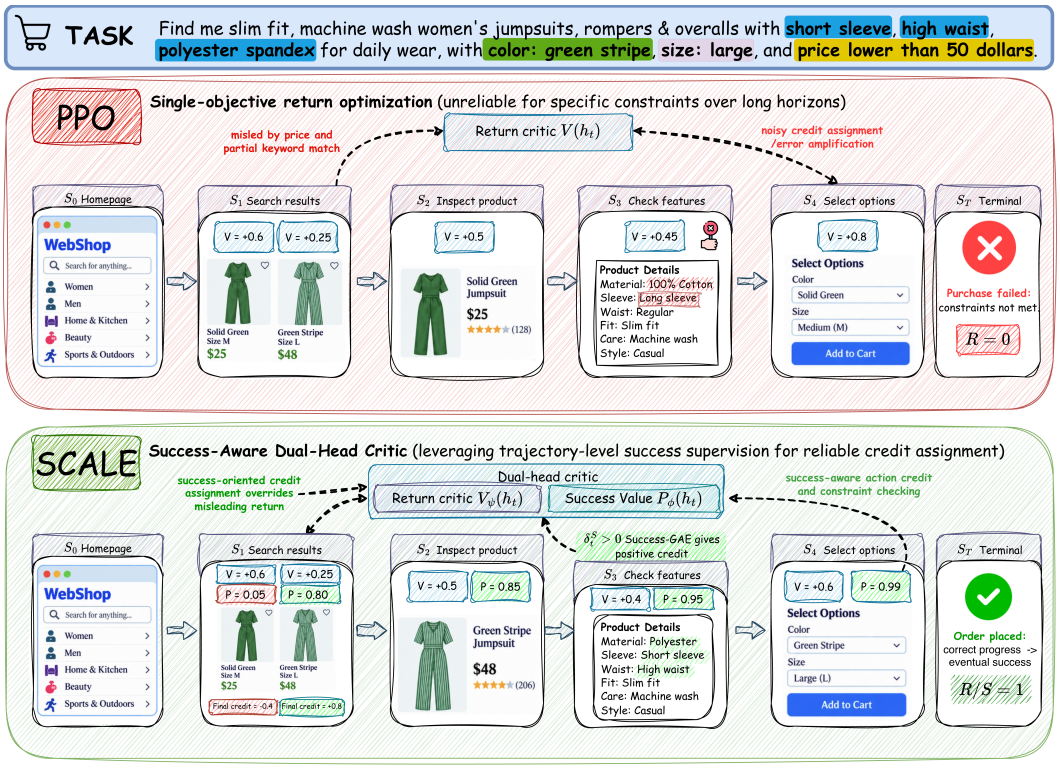

Figure 1: Comparison between PPO and SCALE. PPO derives step credit from changes in predicted return, which can be unreliable when feedback is sparse. An action may receive positive credit even when it lowers the probability of eventual success. SCALE instead uses changes in Success Value to measure whether each action makes task success more or less likely.

To address these challenges, we introduce SCALE (**S**uccess-**C**alibrated **A**dvantage **L**earning and **E**stimation), a success-aware actor–critic framework for long-horizon language agents. SCALE makes eventual task completion an explicit prediction target alongside conventional return estimation, **rather than relying on return alone**. Specifically, the critic retains a return head for expected cumulative reward and adds a Success Value head that estimates the probability of satisfying the benchmark's terminal completion criterion from the current interaction prefix. It then uses changes in this prediction to assign credit to individual environment steps, reflecting whether each action improves eventual success. Because the reliability of this signal can vary during training, SCALE adjusts its contribution according to critic quality. The resulting credit complements the standard return-based advantage while retaining the token-level PPO objective. Sec. 4 details the framework.

We evaluate SCALE on five representative agentic benchmarks spanning text-based interaction and tool use: ALFWorld (Shridhar et al., 2021), WebShop (Yao et al., 2022), ScienceWorld (Wang et al., 2022), BFCL-v4 (Patil et al., 2025), and AppWorld (Trivedi et al., 2024). Across these tasks, SCALE consistently *outperforms* most strong agentic RL baselines and delivers substantial gains in task performance. Beyond end-task performance, We also analyze critic quality and the accuracy of its action credit to understand where these gains come from. Our contributions are threefold:

❶ **Local-credit failure of return critics.** We show that conventional PPO critics become less reliable at distinguishing alternative actions as decisions move farther from terminal supervision.

❷ **Success-Based Credit with Reliability Control.** We introduce SCALE, which explicitly predicts eventual task success from **each interaction prefix** and **converts changes in this probability into multi-step action credit**. It then modulates the contribution of this credit according to **estimated critic reliability**, while retaining the standard token-level PPO objective.

❸ **Empirical validation.** Experiments across five agent benchmarks and three model backbones show improvements in task performance, including gains of up to **4.1** percentage points in success rate. Additional analyses show improved action ranking and support the roles of success-based credit and reliability control.

Table 1: Comparison of representative RL methods for credit assignment. "Gran." denotes credit-assignment granularity. "Critic" indicates whether a learned critic is used. "Reliab." denotes explicit reliability modeling or control, and "Token PO" indicates token-level policy optimization.

Method	Gran.	Critic	Reliab.	Token PO	Credit Signal	Extra Structure
PPO	Step	✓	✗	✓	Discounted return	None
GRPO	Trajectory	✗	✗	✓	Group outcome	Group rollouts
GiGPO	Step	✗	✗	✓	Step relative adv.	Anchor states
VinePPO	Step	✗	✗	✓	Monte Carlo value	Vine branching
Turn-PPO	Turn	✓	✗	✗	Turn return	Turn MDP
CAPO	Action	✓	✗	✗	Action value	Action optimization
SORL	Turn	✓	✓	✗	Turn-level adv.	Off-policy turns
SCALE (ours)	**Step**	✓	✓	✓	**Return + Success**	**None**

2 Related Work

Reinforcement Learning for LLM-based Agents. Reinforcement learning has evolved from preference alignment (Schulman et al., 2017; Ouyang et al., 2022) toward optimization from verifiable task outcomes (Shao et al., 2024), and has recently been extended to interactive language agents. Existing work studies RL for search and tool use (Jin et al., 2025; Feng et al., 2026; Qian et al., 2025), multi-turn environment interaction (Wang et al., 2025b; 2026b), and general agent-training frameworks (Feng et al., 2024; Luo et al., 2025). Scalable RL infrastructures such as verl (Sheng et al., 2025) and ROLL (Wang et al., 2025a) support large-scale rollout generation and environment interaction. Other approaches improve long-horizon learning through planning and exploration (Paglieri et al., 2025; Wang et al., 2026d;c). While these methods improve the training and exploration of language agents, a central challenge remains: *how to convert sparse trajectory-level outcomes into reliable credit for intermediate actions*, particularly for decisions far from terminal supervision.

Credit Assignment and Critic Learning. Existing approaches differ along two dimensions that are central to credit assignment: *where* credit is assigned along a trajectory and *how* that credit is estimated. Critic-free methods derive turn- or step-level signals through comparison, grouping, or structural information (Zeng et al., 2025; Feng et al., 2025; He et al., 2026; Wang et al., 2026d), but often require repeated states, milestones, or other task-dependent structure. Monte Carlo methods estimate local action values from multiple continuations, avoiding a learned critic at the cost of substantial sampling overhead (Kazemnejad et al., 2025). Critic-based methods instead propagate delayed outcomes through learned turn- or action-level values (Zhou et al., 2024; Li et al., 2025b; Wang et al., 2026a; Tan et al., 2026), but reliable local comparisons remain difficult when value estimates are learned from sparse supervision. Recent work further improves robustness by suppressing unreliable updates or controlling the use of uncertain credit signals (Li et al., 2026a;b; Tao et al., 2026). In contrast, **SCALE** explicitly predicts eventual task success from each interaction prefix and uses the estimated reliability of this prediction to control its contribution to environment-step credit.

3 Preliminaries And Analysis

3.1 Problem Setup

We model language agent interaction as a partially observable Markov decision process (POMDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{R}, \Omega, \gamma)$, where $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{O}$ are the state, action, and observation spaces; $\mathcal{P}$, $\mathcal{R}$, and Ω denote the transition kernel, reward function, and observation model; and γ is the return discount factor. At environment step t, the agent observes o_t and conditions on the interaction history $h_t = (o_0, a_0, r_0, \ldots, o_{t-1}, a_{t-1}, r_{t-1}, o_t)$. It generates action $a_t = (y_{t,1}, \ldots, y_{t,L_t})$, where $y_{t,i}$ is token i, according to $\pi_\theta(a_t \mid h_t) = \prod_{i=1}^{L_t} \pi_\theta(y_{t,i} \mid h_t, y_{t,<i})$. The environment transitions after the complete action is executed. An episode of T steps produces $\tau = (o_0, a_0, r_0, \ldots, a_{T-1}, r_{T-1}, o_T)$.

Each benchmark defines a binary success outcome $z \in \{0, 1\}$ under its evaluation protocol (Appendix C). Unlike cumulative return, which aggregates trajectory rewards, z indicates whether the task is completed. For prefix h_t, we define the probability of success as $p^\pi_{\text{succ}}(h_t) = \Pr_\pi(z = 1 \mid h_t)$.

Standard actor–critic methods (Konda & Tsitsiklis, 2000; Mnih et al., 2016) estimate the expected discounted return $V^\pi(h_t) = \mathbb{E}_\pi\left[\sum_{k=t}^{T-1} \gamma^{k-t} r_k \mid h_t\right]$, with a learned critic $V_\psi(h_t)$. Let $d_t \in \{0, 1\}$ indicate episode termination. Given the TD residual $\delta^V_t = r_t + \gamma(1 - d_t)V_\psi(h_{t+1}) - V_\psi(h_t)$, the

conventional return-based advantage is computed using GAE (Schulman et al., 2015):

$$\hat{A}_t^V = \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \delta_{t+l}^V, \tag{1}$$

where $\lambda \in [0, 1]$ is the GAE parameter. We refer to $\hat{A}_t^V$ as the return-based advantage. Our focus is whether these return-based local scores reliably rank alternative actions, rather than only predicting return accurately in aggregate.

3.2 Analysis

We first ask whether a PPO critic can provide reliable *local* credit for long-horizon agent decisions. For policy improvement, the critic must do more than predict return: it must rank alternative actions correctly. This ranking is hard to learn from a limited number of rollouts when feedback is sparse, especially for decisions far from the terminal outcome(Li et al., 2025b; Kazemnejad et al., 2025).

To test this directly, we start from the same interaction prefix, sample different candidate actions, and roll out each action several times. For each prefix h_t, we estimate a Monte Carlo state value $\hat{V}_{\mathrm{MC}}(h_t) = M^{-1} \sum_m G_t^{(m)}$. We further sample K candidate actions and evaluate each with M continuations, yielding $\hat{Q}^R_{\mathrm{MC}}(h_t, a_t^{(j)}) = M^{-1} \sum_m G_t^{(j,m)}$. We measure critic estimation with explained variance and local credit quality with pairwise ordering accuracy between the critic score $\hat{Q}_V(h_t, a_t) = r_t + \gamma(1 - d_t)V_\psi(h_{t+1})$ and the Monte Carlo action values. Prefixes are divided into *Near*, *Mid*, and *Far* according to benchmark-specific tertiles of the remaining horizon $H_t = T - t$.

Observation 1: Critic estimation degrades with horizon. Figure 2(a) shows a consistent drop in explained variance from near to far horizon prefixes across both benchmarks, indicating that return estimation becomes less reliable as decisions move farther from outcome supervision.

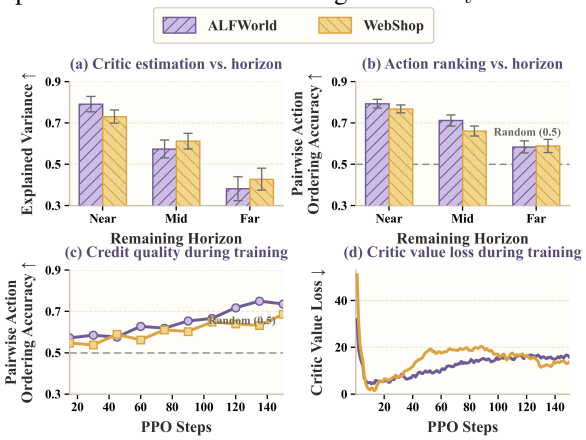

Figure 2: Vanilla PPO critic reliability analysis on ALFWorld and WebShop.

Observation 2: Local action credit degrades more critically. Figure 2(b) shows that pairwise action ordering accuracy decreases with horizon and approaches the random baseline for far horizon. This result shows that the critic's ordering increasingly diverges from Monte Carlo estimates for decisions farther from terminal outcome. Thus, even a critic with nontrivial return-prediction ability may fail to distinguish better actions, consistent with prior findings on learned value models (Kazemnejad et al., 2025).

Observation 3: Credit quality changes throughout training. Figure 2(c) shows that action-ordering accuracy improves overall, but it does not improve steadily across updates. Fig. 2(d) shows that critic value loss initially decreases but later rebounds and fluctuates, showing that critic quality remains unstable throughout training rather than improving steadily with policy updates.(Pan et al., 2026).

Together, these results show a limitation of the standard return critic: its action rankings become less reliable for decisions far from the final outcome and can fluctuate during training. This motivates adding a direct success signal and using it only when the critic is reliable.

4 Methodology

4.1 Overview

As illustrated in Fig. 3, `SCALE` addresses the gap between return prediction and reliable credit assignment in long trajectories. While return critics can implicitly encode task success, this signal may be unreliable for local credit assignment in practice. `SCALE` therefore augments the return critic with an **Success Value** head. **Success-GAE** uses changes in this prediction to compute step credit, and `SCALE` scales this credit according to the **reliability** of the success critic. The resulting step advantage is finally broadcast to all tokens of the corresponding action for token-level PPO optimization.

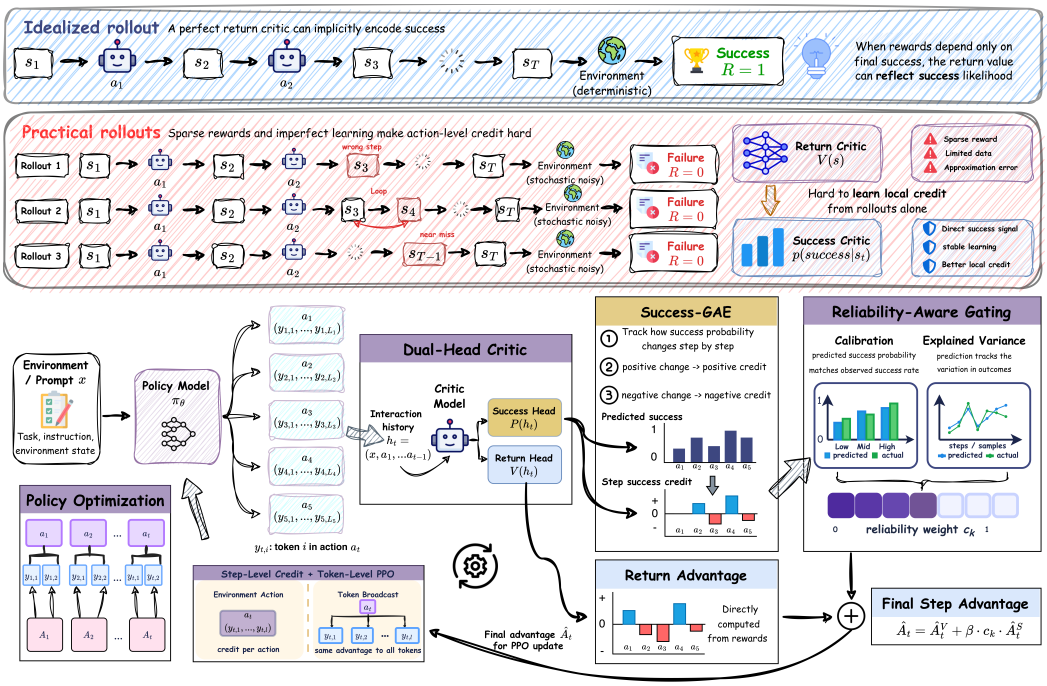

Figure 3: **Overview of SCALE. Top:** A simple rollout example contrasts an ideal return critic with practical training, where sparse supervision and imperfect learning make reliable action-level credit difficult. **Bottom:** SCALE learns success-aware critic signals, converts them into reliable environment-step credit, and uses the resulting advantages for token-level policy optimization.

4.2 Success-Aware Dual-Head Critic

The preceding analysis suggests that accurate return estimation alone may not provide reliable local credit for eventual task success. We therefore augment the conventional return critic with an explicit success-prediction head. Given an interaction prefix h_t, the dual-head critic outputs $\mathcal{C}(h_t) = \big(V_\psi(h_t), P_\phi(h_t)\big)$, where $V_\psi(h_t)$ estimates future return and $P_\phi(h_t)$ denotes the **Success Value** defined in Sec. 3.1. The return head predicts expected future reward. The success head predicts whether the task will eventually be completed from the current prefix. To jointly learn these two signals from collected trajectories, we optimize return regression together with success prediction:

$$\mathcal{L}_{\text{critic}} = \mathbb{E}_\tau \left[\frac{1}{T} \sum_{t=0}^{T-1} \Big(\big(V_\psi(h_t) - \hat{G}_t\big)^2 + \eta_S \, \text{BCE}(P_\phi(h_t), S) \Big) \right]. \tag{2}$$

Here, $\hat{G}_t$ is the return target for prefix h_t, S is the trajectory-level success label, and η_S balances return regression and success prediction. Because each interaction prefix is paired with the eventual task outcome, the success head directly learns to predict eventual task completion from that prefix, rather than relying on this information to emerge implicitly from return fitting. This explicit Success Value will serve as the basis for constructing success-aware environment-step credit in the next stage.

4.3 Success-GAE

The Success Value is defined on interaction prefixes, but policy learning requires credit for the action that moves the agent from h_t to h_{t+1}. We therefore turn successive success predictions into an action-level signal. Let $p_t = P_\phi(h_t)$ and set the terminal prediction to the observed outcome, $p_T = S$. We define the success temporal difference as $\delta_t^S = p_{t+1} - p_t$: an action receives positive credit when it increases the predicted probability of eventual success and negative credit when it decreases it.

A single temporal difference captures only the immediate change in success prospects, whereas long-horizon actions may influence several subsequent steps. We therefore aggregate future success differences in the same spirit as generalized advantage estimation and define **Success-GAE** as

$$\hat{A}_t^S = \sum_{l=0}^{T-t-1} \lambda_S^l \delta_{t+l}^S, \qquad \lambda_S \in [0, 1]. \tag{3}$$

Here, $\hat{A}_t^S$ is the success-oriented advantage for action a_t, δ_{t+l}^S measures subsequent changes in predicted success, and λ_S controls how strongly later changes contribute to the current action. Unlike return-based GAE, Success-GAE does not use the environment discount factor because it tracks changes in task-success probability rather than accumulated reward. The resulting signal assigns credit based on whether intermediate decisions make eventual success more or less likely.

4.4 Reliability-Aware Credit Integration

Reliability Estimation. Because Success-GAE is useful only when the Success Value predictor is reliable, we assess its quality at optimization stage k using the Brier score and explained variance:

$$\mathrm{BS}_k = \frac{1}{N_k} \sum_{n=1}^{N_k} \left(P_\phi(h^{(n)}) - S^{(n)} \right)^2, \tag{4}$$

$$\mathrm{EV}_k^S = 1 - \frac{\mathrm{Var}(S - P_\phi(h))}{\mathrm{Var}(S) + \varepsilon}. \tag{5}$$

Here, BS_k measures absolute probabilistic error, whereas EV_k^S measures whether prediction differences track variation in eventual task success. The two metrics therefore capture complementary aspects of reliability and provide a more complete view of critic quality.

Reliability Weighting. We next convert these metrics into a weight in $[0, 1]$. Let $\bar{S}_k = N_k^{-1} \sum_n S^{(n)}$ denote the success rate in the current batch. A constant predictor $P_\phi(h) = \bar{S}_k$ has Brier score $\bar{S}_k(1 - \bar{S}_k)$, so we use this error as a natural reference for prediction quality and clip explained variance to obtain a comparable reliability scale:

$$q_k^{\mathrm{BS}} = \mathrm{clip}\left(1 - \frac{\mathrm{BS}_k}{\bar{S}_k(1 - \bar{S}_k) + \varepsilon}, 0, 1 \right), \tag{6}$$

$$q_k^{\mathrm{EV}} = \mathrm{clip}\left(\mathrm{EV}_k^S, 0, 1 \right), \qquad c_k = q_k^{\mathrm{BS}} q_k^{\mathrm{EV}}. \tag{7}$$

Thus, q_k^{BS} is positive only when the predictor outperforms the constant baseline, while q_k^{EV} retains only positive explained variance. Their product c_k gives substantial weight to success credit only when both criteria support reliable and informative predictions.

Credit Integration. Finally, we combine the return and weighted success advantages:

$$\hat{A}_t = \hat{A}_t^V + \beta \, c_k \, \hat{A}_t^S, \tag{8}$$

where β controls the contribution of success-based credit. When c_k is small, the update falls back to the return advantage; as reliability improves, success-based credit contributes more.

4.5 Token-Level Policy Optimization

To optimize the autoregressive policy, we assign this advantage to all tokens in the action and retain the standard token-level clipped PPO objective (Schulman et al., 2017). We optimize:

$$\mathcal{L}_{\mathrm{actor}}(\theta) = -\mathbb{E}\left[\frac{1}{\sum_t L_t} \sum_t \sum_{i=1}^{L_t} \min\Big(\rho_{t,i}(\theta) \hat{A}_t, \; \mathrm{clip}(\rho_{t,i}(\theta), 1 - \epsilon, 1 + \epsilon) \, \hat{A}_t \Big) \right]. \tag{9}$$

All tokens within the same action share the environment-step credit $\hat{A}_t$, while the policy update remains standard token-level PPO. Thus, **SCALE** augments the action advantage with reliability-weighted success credit while leaving the PPO actor objective unchanged. Appendix A provides a theoretical analysis of when the added success signal can improve the distinction between actions.

5 Experiment

5.1 Experimental Setup

Benchmarks. We evaluate **SCALE** on five long-horizon agentic benchmarks spanning two major interaction paradigms. ALFWorld (Shridhar et al., 2021), WebShop (Yao et al., 2022), and ScienceWorld (Wang et al., 2022) represent text-based interactive environments, covering embodied household tasks, web navigation, and scientific reasoning. BFCL-v4 (Patil et al., 2025) and AppWorld (Trivedi et al., 2024) evaluate tool-using agents through function calling and application-level execution. These benchmarks let us test **SCALE** across different interaction structures, action spaces, and task horizons. Full benchmark details are provided in Appendix C.

Table 2: **Main results across agentic benchmarks.** We report success rate (%) on ALFWorld, and task score and success rate on ScienceWorld and WebShop. Results are averaged over three runs. **Best** and second-best are highlighted within each base-model block.

Method	ALFWorld							ScienceWorld		WebShop	
	Pick	Look	Clean	Heat	Cool	Pick2	All	Score	Succ	Score	Succ
Closed-Source Models											
GPT-4o	75.3	60.8	31.2	56.7	21.6	49.8	48.0	54.3	45.4	31.8	23.7
Gemini-2.5-Pro	92.8	63.3	62.1	69.0	26.6	58.7	60.3	47.8	36.7	42.5	35.9
Base: Qwen2.5-1.5B-Instruct											
Direct Prompt	5.9	5.5	3.3	9.7	4.2	0.0	4.1	5.9	0.7	23.1	5.2
ReAct	17.4	20.5	15.7	6.2	7.7	2.0	12.8	9.0	1.2	40.1	11.3
Reflexion	35.3	22.2	21.7	13.6	19.4	3.7	21.8	7.1	3.9	55.8	21.9
PPO	64.8	40.5	57.1	60.6	46.4	47.4	54.4	29.3	10.9	73.8	51.5
RLOO	88.3	52.8	71.0	62.8	66.4	56.9	69.7	30.6	18.4	73.9	52.1
GRPO	85.3	53.7	84.5	78.2	59.7	53.5	72.8	31.7	21.1	75.8	56.8
GSPO	91.6	68.4	87.2	84.6	65.8	68.7	79.3	33.9	23.6	79.7	60.9
GiGPO	96.0	76.5	91.8	91.3	71.7	79.5	86.1	35.6	25.8	83.1	65.0
BEACON	**100**	88.2	86.7	**100**	78.9	**92.9**	91.4	**58.9**	45.3	86.1	75.6
ST-PPO	97.4	79.1	93.6	89.7	75.4	83.1	87.7	40.8	31.3	84.8	68.0
CAPO	**100**	84.5	**94.1**	96.2	77.6	89.4	91.2	52.6	40.6	**86.4**	76.6
SCALE (Ours)	**100**	**97.4**	88.9	**100**	**93.3**	82.6	**93.4**	57.8	**46.8**	85.3	**78.1**
Base: Qwen2.5-7B-Instruct											
Direct Prompt	33.4	21.6	19.3	6.9	2.8	3.2	14.8	11.4	4.2	26.4	7.8
ReAct	48.5	35.4	34.3	13.2	18.2	17.6	31.2	17.4	7.8	46.2	19.5
Reflexion	62.0	41.6	44.9	30.9	36.3	23.8	42.7	23.4	11.7	58.1	28.8
PPO	92.3	64.0	92.5	89.5	80.3	68.8	80.4	37.1	24.0	81.4	68.7
RLOO	87.6	78.2	87.3	81.3	71.9	48.9	75.5	57.4	44.6	80.3	65.7
GRPO	90.8	66.1	89.3	74.7	72.5	64.7	77.6	61.8	49.1	79.3	66.1
GSPO	94.6	74.8	93.7	79.6	80.1	72.5	84.5	65.2	51.0	81.7	69.3
GiGPO	97.7	82.7	98.8	83.7	89.3	79.2	90.8	69.2	53.4	84.4	72.8
BEACON	**100**	81.8	96.3	**92.9**	94.7	90.0	94.5	83.7	64.3	87.7	79.7
ST-PPO	**100**	86.4	97.5	88.2	91.0	84.6	92.7	73.8	56.9	86.2	75.4
CAPO	**100**	84.7	**100**	91.6	93.6	88.4	94.5	79.8	61.8	88.9	78.0
SCALE (Ours)	**100**	**100**	**100**	91.3	**100**	**90.5**	**97.4**	**86.7**	**66.5**	**92.7**	**83.8**

Baselines. We compare SCALE against representative prompting and RL baselines. Closed-source references include GPT-4o (OpenAI, 2024) and Gemini-2.5-Pro (Comanici et al., 2025). Prompting baselines include Direct Prompt, ReAct (Yao et al., 2023), and Reflexion (Shinn et al., 2023). RL baselines cover three categories: (1) Standard policy optimization, including PPO (Schulman et al., 2017), RLOO (Ahmadian et al., 2024), GRPO (Shao et al., 2024), and GSPO (Zheng et al., 2025); (2) Fine-grained or structured credit assignment, including GiGPO (Feng et al., 2025), BEACON (Wang et al., 2026d), ST-PPO (Li et al., 2025a), and CAPO (Wang et al., 2026a); and (3) Critic-free sequence-level optimization, represented by SeeUPO (Hu et al., 2026). We use official implementations or closely follow the reported configurations whenever available.

Implementation. Our experiments use Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct (Yang et al., 2024), together with Qwen3-8B (Yang et al., 2025), as backbone models. ALFWorld, WebShop, and ScienceWorld are implemented with verl-agent (Feng et al., 2025), while BFCL-v4 and AppWorld are built on AgentEvolver (Zhai et al., 2025). We retain Qwen2.5 due to more stable agentic performance and infrastructure support, and use the same base model across RL methods for fair comparison. Within each backbone and benchmark setting, all trainable methods share the same initial model, rollout budget, environment interface, and evaluation protocol. Detailed configurations and hyperparameters are provided in Appendix C.

5.2 Main Results

Text-Based Interactive Benchmarks. Table 2 reports results on ALFWorld, ScienceWorld, and WebShop. With Qwen2.5-1.5B-Instruct, SCALE reaches success rates of 93.4% on ALFWorld, 46.8% on ScienceWorld, and 78.1% on WebShop, exceeding the strongest RL baselines by 2.0, 1.5, and 1.5 percentage points, respectively. With Qwen2.5-7B-Instruct, SCALE further improves to 97.4%, 66.5%, and 83.8%, with gains of 2.9, 2.2, and 4.1 percentage points over the strongest baselines.

These gains appear across all three text-interaction benchmarks and both model sizes, indicating that the benefit of SCALE is not limited to a single environment or backbone scale.

Tool-Use Benchmarks. Table 3 evaluates SCALE on AppWorld and BFCL-v4 with Qwen2.5-7B-Instruct and Qwen3-8B. SCALE achieves the best average performance across the two benchmarks with both backbones, reaching avg@4/pass@4 scores of 43.57/56.01 with Qwen2.5-7B and 50.44/62.47 with Qwen3-8B. On BFCL-v4, SCALE obtains 44.25/55.00 and 51.75/57.00, respectively, outperforming all compared RL baselines. On AppWorld, SCALE achieves the highest avg@4 with Qwen2.5-7B and the highest pass@4 with Qwen3-8B, while remaining close to SeeUPO on the other metric. Overall, the gains extend from text-interaction benchmarks to tool-use environments.

Table 3: **Results on agent tool-use benchmarks.** We report *avg@4* and *pass@4*. **Best** and second-best are highlighted within each base-model block.

Method	AppWorld		BFCL-v4		Avg.	
	avg@4	pass@4	avg@4	pass@4	avg@4	pass@4
Qwen2.5-7B-Instruct						
Base	12.63	24.56	18.00	27.00	15.32	25.78
+PPO	27.36	42.11	34.75	43.00	31.06	42.56
+GRPO	30.70	45.61	37.50	46.00	34.10	45.81
+GSPO	25.00	40.35	40.25	47.00	32.63	43.68
+GiGPO	35.44	52.63	43.75	49.00	39.60	50.82
+SeeUPO	40.74	**57.89**	42.50	50.00	41.62	53.95
+SCALE	**42.89**	57.02	**44.25**	**55.00**	**43.57**	**56.01**
Qwen3-8B						
Base	13.63	28.07	31.25	38.00	22.44	33.04
+PPO	26.68	43.86	36.25	44.00	31.47	43.93
+GRPO	31.71	49.12	37.75	47.00	34.73	48.06
+GSPO	25.96	43.86	39.25	48.00	32.61	45.93
+GiGPO	40.09	57.89	47.25	54.00	43.67	55.95
+SeeUPO	**50.39**	65.79	46.25	54.00	48.32	59.82
+SCALE	49.12	**68.42**	**51.75**	**57.00**	**50.44**	**62.47**

5.3 Framework Analysis

Episode Length. Figure 4 compares the average episode length on the three text-interaction benchmarks. SCALE reduces the number of environment steps from 27.6 to 19.1 on ALFWorld and from 19.7 to 13.2 on ScienceWorld, while remaining close to GiGPO on WebShop (6.9 vs. 6.8). Together with the task results, this suggests that SCALE achieves better outcomes with fewer interaction steps, avoiding unnecessarily long action sequences rather than relying on extended exploration.

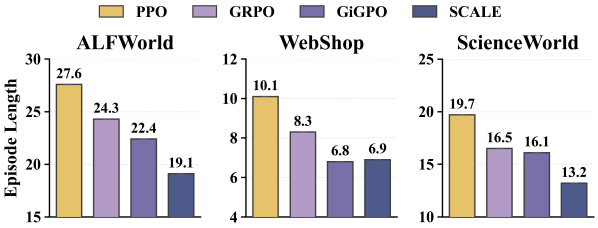

Figure 4: **Episode length on text-interaction benchmarks.** We report the average number of environment steps per episode at the final checkpoint on ALFWorld, WebShop, and ScienceWorld.

Training Efficiency. Figure 5 compares wall-clock time per training iteration on BFCL-v4 and AppWorld. Although SCALE adds a success head and reliability estimation, its training cost remains close to GRPO and GiGPO and is much lower than SeeUPO. This provides the performance gains without a large increase in training cost, suggesting that the additional critic computation introduces only modest practical overhead.

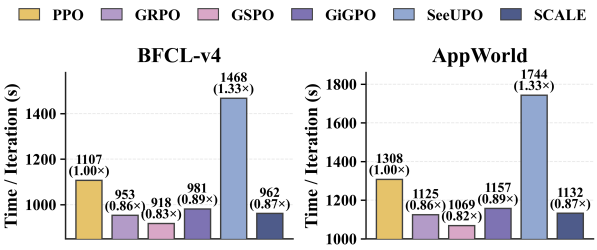

Figure 5: **Training efficiency on BFCL-v4 and AppWorld.** We report wall-clock time per iteration with Qwen3-8B. Relative cost is normalized to PPO.

Critic Learning. Figure 6 shows that SCALE achieves higher critic explained variance than PPO across all five benchmarks, indicating that its critic better captures variation in future returns. Its critic loss also decreases more smoothly and remains lower, whereas PPO shows larger fluctuations or later increases in several settings. These results suggest that adding the success signal helps the critic learn more stable and informative value estimates, with the same pattern observed across interaction settings.

Training Performance. The improved critic dynamics are accompanied by stronger policy learning. Across ALFWorld, WebShop, and ScienceWorld, SCALE reaches higher training success rates than PPO, while on BFCL-v4 and AppWorld it achieves consistently higher *avg@4*. The performance gap

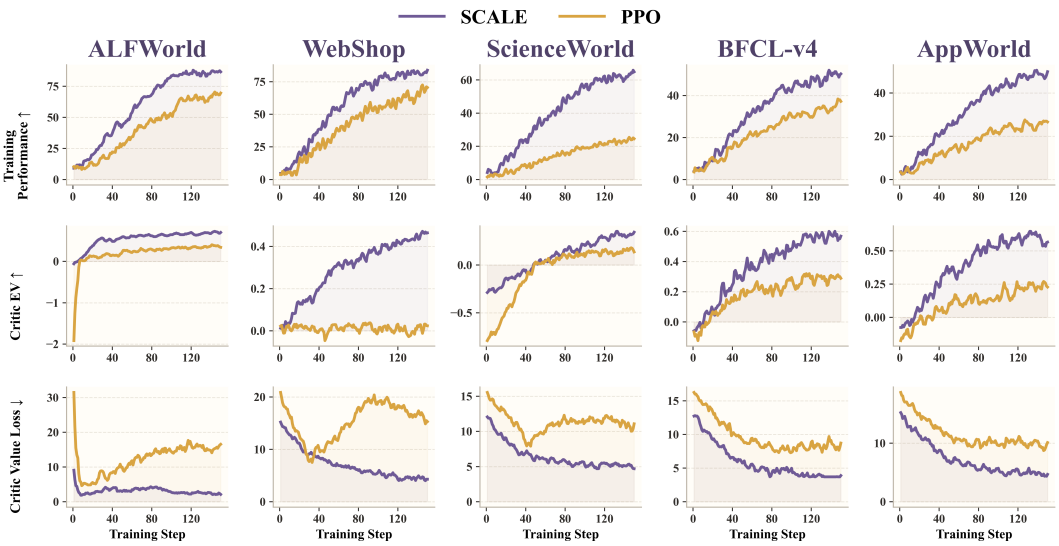

Figure 6: **Training dynamics across five agent benchmarks. Top:** Task performance during training, reported as success rate for ALFWorld, WebShop, and ScienceWorld, and *avg@4* for BFCL-v4 and AppWorld. **Middle:** Critic explained variance during training. **Bottom:** Critic value loss during training. Compared with PPO, SCALE achieves higher explained variance and generally lower, smoother critic loss across the five benchmarks.

also tends to widen over training, suggesting that better critic learning translates into more effective long-horizon policy optimization rather than merely better value fitting.

Ablation Study. Table 4 shows that removing any of the tested components lowers the performance of SCALE. Removing the Success Value head lowers *avg@4* from 51.75 to 43.25 on BFCL-v4 and from 49.12 to 39.91 on AppWorld, while removing the return head causes larger drops to 29.75 and 20.18. Replacing multi-step success credit with one-step TD also degrades performance to 39.75 and 31.14. Finally, using only Brier score or explained variance remains below the full method on both benchmarks, supporting their combined use for reliability-aware integration. Overall, the results show that return estimation, success-based credit, and reliability weighting each play a distinct role in the full framework.

Table 4: **Ablation study of SCALE with Qwen3-8B.** We report *avg@4* (%) on BFCL-v4 and AppWorld.

Variant	BFCL-v4	AppWorld
SCALE	**51.75**	**49.12**
Success-Aware Critic		
w/o Success Value Head	43.25	39.91
w/o Return Value Head	29.75	20.18
Success Credit Estimation		
w/ One-Step Success TD	39.75	31.14
w/o Success Credit	46.25	43.97
Reliability-Aware Integration		
w/o Reliability Weighting	48.00	45.83
w/ Brier Score Only	49.25	47.04
w/ Explained Variance Only	50.00	47.81
PPO	36.25	26.68

6 Conclusion

We introduced SCALE for credit assignment in long-horizon language agents. Our analysis shows that a critic can retain aggregate return-prediction quality while becoming less reliable at ranking actions, especially for decisions far from terminal supervision. SCALE addresses this gap by predicting eventual task success, converting changes in this probability into multi-step action credit, and controlling its contribution according to predictor reliability. Across five agent benchmarks and three model backbones, SCALE improves task performance, while maintaining modest training overhead. These results show that improving local credit quality can benefit long-horizon policy learning without changing the underlying token-level PPO objective. Future work could explore incorporating richer signals into critic learning to improve reliability under long-horizon supervision.

AI USE STATEMENT

Generative AI tools were used only to improve the readability and linguistic presentation of author-written text. They were not used to develop the methodology, formulate mathematical claims, design experiments, generate or process data. All suggested edits were manually reviewed by the authors, who take full responsibility for the final content of this submission.

ETHICS STATEMENT

This work does not involve human participants, personally identifiable information, or private and sensitive data. All datasets, software, and experimental procedures were used in accordance with their applicable licenses and research policies. To the best of the authors' knowledge, this work does not raise additional material ethical concerns.

REPRODUCIBILITY STATEMENT

We provide the information required to reproduce our results throughout the main paper and supplementary materials. The benchmark datasets are described in Appendix C in detail. Training pipeline includes framework details and evaluation protocols are reported in Appendix B.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.662.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Ruo Yu Tao, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. Textworld: A learning environment for text-based games. *arXiv preprint arXiv:1806.11532*, 2018.

Yifeng Ding, Hung Le, Songyang Han, Kangrui Ruan, Zhenghui Jin, Varun Kumar, Zijian Wang, and Anoop Deoras. Empowering multi-turn tool-integrated agentic reasoning with group turn policy optimization. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 42409–42423. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1962.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. ReTool: Reinforcement learning for strategic tool use in LLMs. In *International Conference on Learning Representations*, 2026.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems*, volume 38, pp. 51797–51830, 2025. doi: 10.52202/085713-1544.

Peiyuan Feng, Yichen He, Guanhua Huang, Yuan Lin, Hanchong Zhang, Yuchen Zhang, and Hang Li. AGILE: A novel reinforcement learning framework of LLM agents. *arXiv preprint arXiv:2405.14751*, 2024.

Shuo He, Lang Feng, Qi Wei, Xin Cheng, Lei Feng, and Bo An. Hierarchy-of-groups policy optimization for long-horizon agentic tasks. In *International Conference on Learning Representations*, 2026.

Zhenyu Hou, Yujiang Li, Jie Tang, and Yuxiao Dong. Single-rollout asynchronous optimization for agentic reinforcement learning. *arXiv preprint arXiv:2607.07508*, 2026.

Tianyi Hu, Qingxu Fu, Yanxi Chen, Zhaoyang Liu, and Bolin Ding. SeeUPO: Sequence-level agentic-RL with convergence guarantees. *arXiv preprint arXiv:2602.06554*, 2026.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O. Arık, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Proceedings of the 2nd Conference on Language Modeling*, 2025.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 29557–29590. PMLR, 2025.

Vijay R. Konda and John N. Tsitsiklis. Actor-critic algorithms. In *Advances in Neural Information Processing Systems*, volume 12, 2000.

Chenliang Li, Adel Elmahdy, Alex Boyd, Zhongruo Wang, Alfredo Garcia, Parminder Bhatia, Taha Kass-Hout, Cao Xiao, and Mingyi Hong. ST-PPO: Stabilized off-policy proximal policy optimization for multi-turn agents training. *arXiv preprint arXiv:2511.20718*, 2025a.

Chenliang Li, Adel Elmahdy, Alex Boyd, Zhongruo Wang, Siliang Zeng, Alfredo Garcia, Parminder Bhatia, Taha Kass-Hout, Cao Xiao, and Mingyi Hong. Stabilizing off-policy training for long-horizon LLM agent via turn-level importance sampling and clipping-triggered normalization. *arXiv preprint arXiv:2511.20718*, 2026a.

Junbo Li, Peng Zhou, Rui Meng, Meet P. Vadera, Lihong Li, and Yang Li. Turn-PPO: Turn-level advantage estimation with PPO for improved multi-turn RL in agentic LLMs. *arXiv preprint arXiv:2512.17008*, 2025b.

Yuanfan Li, Qi Zhou, Wenjing Duan, and Lu Chen. When denser credit is not enough: Evidence-calibrated policy optimization for long-horizon LLM agent training. *arXiv preprint arXiv:2606.05885*, 2026b.

Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Siyun Zhao, Dongsheng Li, Luna K. Qiu, and Yuqing Yang. Agent lightning: Train any ai agents with reinforcement learning. *arXiv preprint arXiv:2508.03680*, 2025.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1928–1937. PMLR, 2016.

OpenAI. GPT-4o system card. OpenAI, 2024. Published August 8, 2024.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

Davide Paglieri, Bartłomiej Cupiał, Jonathan Cook, Ulyana Piterbarg, Jens Tuyls, Edward Grefenstette, Jakob Nicolaus Foerster, Jack Parker-Holder, and Tim Rocktäschel. Learning when to plan: Efficiently allocating test-time compute for LLM agents. *arXiv preprint arXiv:2509.03581*, 2025.

Chengjun Pan, Shichun Liu, Jiahang Lin, Dingwei Zhu, Jiazheng Zhang, Shihan Dou, Songyang Gao, Zhenhua Han, Binghai Wang, Rui Zheng, Xuanjing Huang, Tao Gui, and Yansong Feng. EVPO: Explained variance policy optimization for adaptive critic utilization in LLM post-training. *arXiv preprint arXiv:2604.19485*, 2026.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 2025.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. ToolRL: Reward is all tool learning needs. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3524.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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

Noah Shinn, Federico Cassano, Ashwin Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10740–10749, 2020.

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

Hui-Ze Tan, Xiao-Wen Yang, Hao Chen, Jie-Jing Shao, Yi Wen, Yuteng Shen, Weihong Luo, Xiku Du, Lan-Zhe Guo, and Yu-Feng Li. Hindsight credit assignment for long-horizon LLM agents. *arXiv preprint arXiv:2603.08754*, 2026.

Leitian Tao, Baolin Peng, Wenlin Yao, Tao Ge, Hao Cheng, Mike Hang Wang, Jianfeng Gao, and Sharon Li. TRACE: Turn-level reward assignment via credit estimation for long-horizon agents. *arXiv preprint arXiv:2607.13988*, 2026.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.850.

Daoyu Wang, Qingchuan Li, Mingyue Cheng, Jie Ouyang, Shuo Yu, Chunli Liu, Shijin Wang, Qi Liu, and Enhong Chen. CAPO: Critic-guided action-aligned policy optimization for advancing LLM agent capabilities. *arXiv preprint arXiv:2604.18401*, 2026a.

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*, pp. 11279–11298. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.emnlp-main.775.

Weixun Wang, Shaopan Xiong, Gengru Chen, Wei Gao, Sheng Guo, Yancheng He, Ju Huang, Jiaheng Liu, Zhendong Li, Xiaoyang Li, et al. Reinforcement learning optimization for large-scale learning: An efficient and user-friendly scaling library. *arXiv preprint arXiv:2506.06122*, 2025a.

Zenus Wang, Chi Gui, Xing Jin, Qineng Wang, Licheng Liu, Kangrui Wang, Shiqi Chen, Linjie Li, Zhengyuan Yang, Pingyue Zhang, Yiping Lu, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. Ragen-2: Reasoning collapse in agentic rl, 2026b.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Fei-Fei Li, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025b.

Zixuan Wang, Yanrui Miao, Zhengxi Lu, Teng Pan, Yiwen Qiu, Hongxing Li, Peng Qiu, Ruiqing Zhang, and Yongliang Shen. Agent-g^2: Gaussian guidance for agentic reinforcement learning. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, 2026c. Main Conference.

Zixuan Wang, Yuchen Yan, Hongxing Li, Teng Pan, Dingming Li, Ruiqing Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Milestone-guided policy learning for long-horizon language agents. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026d.

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

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

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

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

Siliang Zeng, Quan Wei, William Brown, Oana Frunza, Yuriy Nevmyvaka, and Mingyi Hong. Reinforcing multi-turn reasoning in LLM agents via turn-level credit assignment. *arXiv preprint arXiv:2505.11821*, 2025.

Yunpeng Zhai, Shuchang Tao, Cheng Chen, Anni Zou, Ziqian Chen, Qingxu Fu, Shinji Mai, Li Yu, Jiaji Deng, Zouying Cao, et al. Agentevolver: Towards efficient self-evolving agent system. *arXiv preprint arXiv:2511.10395*, 2025.

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. *arXiv preprint arXiv:2507.18071*, 2025.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. ArCHer: Training language model agents via hierarchical multi-turn RL. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62178–62209. PMLR, 2024.

A THEORETICAL ANALYSIS OF SUCCESS-AWARE CREDIT

This section analyzes the success-aware credit used in **SCALE**. Our goal is not to prove convergence of token-level PPO. Instead, we study two questions: **why Success-GAE can provide useful credit for intermediate actions**, and **when weighting this credit by critic reliability can improve action discrimination**.

A.1 LOCAL SUCCESS-AWARE CREDIT

Consider an interaction prefix h, a continuation policy π, and K possible actions $\{a_1, \dots, a_K\}$. After choosing a_i, the agent follows π until the episode ends and observes $S \in \{0, 1\}$. We define

$$\mu_i = \Pr(S = 1 \mid h, a_i; \pi). \tag{10}$$

Thus, assigning credit at h amounts to estimating and comparing the success probabilities $\{\mu_i\}_{i=1}^K$.

Let $h_0 = h$, $\widehat{p}_j = P_\phi(h_j)$, $\widehat{p}_H = S$, and $\lambda = \lambda_S$. Success-GAE is

$$\widehat{A}^S = \sum_{j=0}^{H-1} \lambda^j (\widehat{p}_{j+1} - \widehat{p}_j). \tag{11}$$

All actions considered at h share the same value $\widehat{p}_0$. We therefore study

$$\widehat{Z} = \widehat{A}^S + \widehat{p}_0, \tag{12}$$

which gives the same ordering over actions as $\widehat{A}^S$.

Proposition 1 (Properties of Success-GAE)**.** *The score $\widehat{Z}$ can be written as*

$$\widehat{Z} = (1 - \lambda) \sum_{j=1}^{H-1} \lambda^{j-1} \widehat{p}_j + \lambda^{H-1} S. \tag{13}$$

Hence $\widehat{Z} \in [0, 1]$.

Let

$$p_j^\star = \Pr(S = 1 \mid h_j; \pi), \tag{14}$$

and let $Z^\star$ be the score in Eq. equation 13 with $\widehat{p}_j$ replaced by $p_j^\star$. For every action a_i,

$$\mathbb{E}[Z^\star \mid h, a_i] = \mu_i, \qquad \mathrm{Var}(Z^\star \mid h, a_i) \le \mu_i(1 - \mu_i). \tag{15}$$

If

$$|\widehat{p}_j - p_j^\star| \le \varepsilon_S, \qquad 1 \le j < H, \tag{16}$$

then

$$|\widehat{Z} - Z^\star| \le d_S := (1 - \lambda^{H-1})\varepsilon_S \le \varepsilon_S, \tag{17}$$

and therefore

$$\left| \mathbb{E}[\widehat{Z} \mid h, a_i] - \mu_i \right| \le d_S. \tag{18}$$

Proof. We first derive Eq. equation 13. Expanding Success-GAE gives

$$\begin{aligned} \widehat{A}^S &= \sum_{j=0}^{H-1} \lambda^j (\widehat{p}_{j+1} - \widehat{p}_j) \\ &= -\widehat{p}_0 + \sum_{j=1}^{H-1} (\lambda^{j-1} - \lambda^j) \widehat{p}_j + \lambda^{H-1} S \\ &= -\widehat{p}_0 + (1 - \lambda) \sum_{j=1}^{H-1} \lambda^{j-1} \widehat{p}_j + \lambda^{H-1} S. \end{aligned} \tag{19}$$

Adding $\widehat{p}_0$ gives Eq. equation 13. For $\lambda \in [0, 1]$, all coefficients are nonnegative and

$$(1-\lambda)\sum_{j=1}^{H-1}\lambda^{j-1} + \lambda^{H-1} = 1. \tag{20}$$

Thus, $\widehat{Z}$ is a convex combination of values in $[0, 1]$, so $\widehat{Z} \in [0, 1]$. Since $\widehat{p}_0$ is the same for all actions considered at h, adding it does not change their ordering.

We next consider the case in which the critic predicts the true success probability. Let $\mathcal{F}_j$ contain the information available at h_j, and define

$$M_0 = \mu_i, \qquad M_j = \mathbb{E}[S \mid \mathcal{F}_j] = p^\star_j, \quad 1 \le j \le H. \tag{21}$$

At termination, $M_H = S$. By the tower property,

$$\begin{aligned}\mathbb{E}[M_{j+1} \mid \mathcal{F}_j] &= \mathbb{E}[\mathbb{E}[S \mid \mathcal{F}_{j+1}] \mid \mathcal{F}_j] \\ &= \mathbb{E}[S \mid \mathcal{F}_j] = M_j.\end{aligned} \tag{22}$$

Therefore, $(M_j)_{j=0}^{H}$ is a martingale, and

$$\mathbb{E}[M_j] = \mu_i \tag{23}$$

for every j. Applying this result to Eq. equation 13 gives

$$\begin{aligned}\mathbb{E}[Z^\star] &= (1-\lambda)\sum_{j=1}^{H-1}\lambda^{j-1}\mathbb{E}[M_j] + \lambda^{H-1}\mathbb{E}[M_H] \\ &= \mu_i\left[(1-\lambda)\sum_{j=1}^{H-1}\lambda^{j-1} + \lambda^{H-1}\right] = \mu_i.\end{aligned} \tag{24}$$

We now bound the variance. Define

$$D_j = M_j - M_{j-1}, \qquad j = 1, \dots, H. \tag{25}$$

Since $M_j = M_0 + \sum_{k=1}^{j} D_k$, Eq. equation 13 implies

$$Z^\star - \mu_i = \sum_{j=1}^{H}\lambda^{j-1}D_j. \tag{26}$$

For $j < k$, D_j is determined by $\mathcal{F}_{k-1}$, while

$$\mathbb{E}[D_k \mid \mathcal{F}_{k-1}] = 0. \tag{27}$$

Hence

$$\mathbb{E}[D_j D_k] = \mathbb{E}[D_j\mathbb{E}[D_k \mid \mathcal{F}_{k-1}]] = 0. \tag{28}$$

The increments are therefore orthogonal, and

$$\begin{aligned}\mathrm{Var}(Z^\star) &= \sum_{j=1}^{H}\lambda^{2(j-1)}\mathbb{E}[D_j^2] \\ &\le \sum_{j=1}^{H}\mathbb{E}[D_j^2] = \mathrm{Var}\left(\sum_{j=1}^{H}D_j\right) \\ &= \mathrm{Var}(S - \mu_i).\end{aligned} \tag{29}$$

Conditioned on h and a_i, S is Bernoulli with mean μ_i. Therefore,

$$\mathrm{Var}(Z^\star) \le \mu_i(1-\mu_i). \tag{30}$$

Finally, we consider an imperfect critic. The terminal value is exact because $\widehat{p}_H = p^\star_H = S$. Therefore,

$$\begin{aligned}|\widehat{Z} - Z^\star| &= \left|(1-\lambda)\sum_{j=1}^{H-1}\lambda^{j-1}(\widehat{p}_j - p^\star_j)\right| \\ &\le (1-\lambda)\sum_{j=1}^{H-1}\lambda^{j-1}|\widehat{p}_j - p^\star_j| \\ &\le (1-\lambda)\sum_{j=1}^{H-1}\lambda^{j-1}\varepsilon_S \\ &= (1-\lambda^{H-1})\varepsilon_S = d_S. \end{aligned} \tag{31}$$

Because $1-\lambda^{H-1} \le 1$, we also have $d_S \le \varepsilon_S$. Combining this bound with $\mathbb{E}[Z^\star] = \mu_i$ gives

$$\begin{aligned}\left|\mathbb{E}[\widehat{Z}] - \mu_i\right| &= \left|\mathbb{E}[\widehat{Z} - Z^\star]\right| \\ &\le \mathbb{E}[|\widehat{Z} - Z^\star|] \le d_S. \end{aligned} \tag{32}$$

□

As a direct consequence, if $\mu_i - \mu_j > 2d_S$, then $\mathbb{E}[\widehat{Z} \mid h, a_i] > \mathbb{E}[\widehat{Z} \mid h, a_j]$. Thus, Success-GAE preserves the expected ordering of two actions whenever their true success probabilities differ by more than the error allowed by the critic. This connects Proposition 1 directly to the pairwise ordering metric in Sec. 3.2.

A.2 Reliability-Weighted Credit Integration

We next study what happens when return credit and success credit are combined. Let $Y_i \in [0,1]$ denote the score obtained from the return critic. Suppose

$$\mathbb{E}[Y_i] = \nu_i + b^V_i, \qquad |b^V_i| \le \varepsilon_V, \qquad \mathrm{Std}(Y_i) \le \sigma_V, \tag{33}$$

where ν_i is the value that Y_i would estimate without critic error.

Let $\alpha \ge 0$ specify how much success credit is added. In **SCALE**, this quantity corresponds to the scale set by βc_k. For analysis, define

$$X_i = \frac{Y_i + \alpha\widehat{Z}_i}{1+\alpha}, \qquad u_i = \frac{\nu_i + \alpha\mu_i}{1+\alpha}. \tag{34}$$

The denominator is shared across actions and therefore does not affect their ordering.

Proposition 1 gives

$$|\mathbb{E}[X_i] - u_i| \le B_\alpha := \frac{\varepsilon_V + \alpha d_S}{1+\alpha}. \tag{35}$$

Suppose the standard deviation of $Z^\star$ is at most σ_S. We do not require Y_i and $\widehat{Z}_i$ to be independent. By the triangle inequality for standard deviations,

$$\mathrm{Var}(X_i) \le v_\alpha := \min\left\{\frac{1}{4}, \left(\frac{\sigma_V + \alpha(\sigma_S + d_S)}{1+\alpha}\right)^2\right\}. \tag{36}$$

The bound B_α controls how far the expected score can move because of critic error. The bound v_α controls how much the observed score can vary across rollouts.

A.3 Finite-Time Analysis of Action Selection

We use a UCB procedure only to study the consequence of these two bounds. UCB is not part of **SCALE**. Let

$$u_\star = \max_i u_i, \qquad \Delta_i = u_\star - u_i. \tag{37}$$

For convenience, define

$$g_i = \Delta_i - 2B_\alpha. \tag{38}$$

Here, g_i is simply the remaining difference between action i and the best action after allowing for critic bias on both scores.

Theorem 1 (Bound on local action selection)**.** *Let $N \geq K$, $\delta \in (0, 1)$, and*

$$L = \log \frac{2KN}{\delta}. \tag{39}$$

Suppose that, during the part of training considered here, observations obtained from each action are i.i.d. and satisfy Eqs. equation 35–equation 36. If $g_i > 0$ for every suboptimal action, then with probability at least $1 - \delta$,

$$R_N \leq \sum_{i:\Delta_i>0} \Delta_i \left[2 + \frac{32 v_\alpha L}{g_i^2} + \frac{8L}{3g_i}\right], \tag{40}$$

where

$$R_N = \sum_{i:\Delta_i>0} \Delta_i T_i(N). \tag{41}$$

Proof. Bernstein's inequality implies that, with probability at least $1 - \delta$, the empirical mean for every action and every sample count up to N lies within

$$r(m) = \sqrt{\frac{2v_\alpha L}{m}} + \frac{2L}{3m} \tag{42}$$

of its expectation.

Suppose a suboptimal action i is selected after it has been observed m times. Its upper confidence bound must then be at least as large as that of the best action. Using Eq. equation 35 for both actions gives

$$\Delta_i \leq 2B_\alpha + 2r(m), \tag{43}$$

and therefore

$$g_i \leq 2r(m). \tag{44}$$

However, if

$$m > \max\left\{\frac{32 v_\alpha L}{g_i^2}, \frac{8L}{3g_i}\right\}, \tag{45}$$

then each term in $r(m)$ is smaller than $g_i/4$, which gives $2r(m) < g_i$. This contradicts the condition required for action i to be selected. Hence

$$T_i(N) \leq 2 + \frac{32 v_\alpha L}{g_i^2} + \frac{8L}{3g_i}. \tag{46}$$

Multiplying by Δ_i and summing over all suboptimal actions gives Eq. equation 40. □

The theorem shows why critic bias matters for action discrimination. If $\Delta_i \leq 2B_\alpha$, the critic error can be large enough to reverse the expected ordering of the two actions, even if more rollouts are collected.

When does success-aware credit help? To compare the combined score with return credit alone, consider the case in which both signals target success:

$$\nu_i = \mu_i \qquad \text{for all } i. \tag{47}$$

If

$$d_S < \varepsilon_V, \qquad \sigma_S + d_S < \sigma_V, \tag{48}$$

then for every $\alpha > 0$,

$$B_\alpha < \varepsilon_V \tag{49}$$

and

$$\Delta_i - 2B_\alpha > \Delta_i - 2\varepsilon_V. \tag{50}$$

The variance bound in Eq. equation 36 also does not increase. Thus, when the success signal has smaller error and lower variation than the return signal, adding it leaves a larger separation between the expected scores of different actions and tightens the bound in Theorem 1.

A.4 Connection to Reliability Metrics

We finally relate the reliability statistics in **SCALE** to the error terms above. Let

$$p^\star(h) = \mathbb{E}[S \mid h], \tag{51}$$

where h is drawn from the distribution $\mathcal{D}$ of prefixes visited by the current policy. The population Brier score satisfies

$$\mathrm{BS} = \underbrace{\mathbb{E}_{\mathcal{D}}\left[(P_\phi(h) - p^\star(h))^2\right]}_{\mathcal{E}_S} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[p^\star(h)(1 - p^\star(h))\right]}_{\text{outcome uncertainty}}. \tag{52}$$

Here, $\mathcal{E}_S$ is the mean-squared error in the predicted success probability. Since both terms are nonnegative,

$$\mathcal{E}_S \le \mathrm{BS}. \tag{53}$$

A lower Brier score therefore gives a smaller upper bound on this prediction error.

We next relate this average error to prefixes reached after a particular action. Let $\mathcal{D}_{i,j}$ be the distribution of h_j after choosing action a_i. Suppose

$$\frac{d\mathcal{D}_{i,j}}{d\mathcal{D}} \le C. \tag{54}$$

Then

$$\begin{aligned}\mathbb{E}_{\mathcal{D}_{i,j}}\left[|P_\phi - p^\star|\right] &\le \sqrt{\mathbb{E}_{\mathcal{D}_{i,j}}\left[(P_\phi - p^\star)^2\right]} \\ &\le \sqrt{C\mathcal{E}_S} \le \sqrt{C\,\mathrm{BS}}.\end{aligned} \tag{55}$$

Using the weights in Eq. equation 13, we obtain

$$\left|\mathbb{E}[\widehat{Z}_i] - \mu_i\right| \le (1 - \lambda^{H-1})\sqrt{C\,\mathrm{BS}}. \tag{56}$$

Thus, when $\mathcal{D}$ covers the prefixes reached after each candidate action, the Brier score also bounds the error in the expected success score.

Explained variance describes a different part of critic quality. It is defined as

$$\mathrm{EV}^S = 1 - \frac{\mathrm{Var}(S - P_\phi(h))}{\mathrm{Var}(S) + \varepsilon}. \tag{57}$$

If

$$m = \mathbb{E}[S - P_\phi(h)], \tag{58}$$

then

$$\mathrm{Var}(S - P_\phi(h)) = \mathrm{BS} - m^2. \tag{59}$$

The two statistics are therefore related, but they describe the error from different perspectives. Brier score measures squared prediction error, whereas explained variance measures how much residual variation remains relative to the variation in S.

For this reason, we use both quantities when computing $c_k = q_k^{\mathrm{BS}} q_k^{\mathrm{EV}}$. We do not claim that this product is an optimal weighting rule. Its role is to reduce the contribution of the success signal when the critic performs poorly according to either statistic. With $\alpha = \beta c_k$, the part of the bias contributed by the success score is at most

$$\frac{\beta c_k}{1 + \beta c_k} d_S, \tag{60}$$

which approaches zero as $c_k \to 0$.

Summary. The analysis gives three results. First, Success-GAE estimates eventual success without changing its expectation when the Success Critic is exact, and its error remains bounded when the critic is approximate. Second, this error determines when the expected ordering of two actions is preserved. Third, if the success signal is more accurate and less variable than the return signal, adding it can improve the separation between actions and tighten the bound on local action selection.

B Details of the Benchmarks

We evaluate **SCALE** on five benchmarks spanning household interaction, web navigation, scientific experimentation, function calling, and application-level tool use.

ALFWorld. ALFWorld (Shridhar et al., 2021) is a text-based environment derived from ALFRED (Shridhar et al., 2020) and TextWorld (Côté et al., 2018). An agent receives natural-language observations and must complete household tasks by navigating rooms and manipulating objects through textual commands. The benchmark includes tasks such as picking up, cleaning, heating, cooling, and examining objects. We use the standard benchmark split and report task success rate.

WebShop. WebShop (Yao et al., 2022) simulates online shopping over a large collection of real products. Given a natural-language instruction, an agent searches for products, navigates product pages, selects options, and attempts to purchase an item that satisfies the requested attributes. The environment provides a graded task score based on how well the selected product matches the instruction, together with a binary success signal. We report both task score and success rate.

ScienceWorld. ScienceWorld (Wang et al., 2022) contains interactive science tasks in which an agent performs experiments and manipulates objects through natural-language actions. The tasks cover topics from physics, chemistry, and biology and require several dependent steps before the goal can be completed. We evaluate across the benchmark tasks and report both task score and success rate.

BFCL-v4. BFCL-v4 (Patil et al., 2025) evaluates the ability of language models to use external functions correctly. The agent must interpret a user request, choose appropriate functions, provide valid arguments, and use function outputs when further actions are required. This setting tests whether a policy can make reliable tool-use decisions over multiple interaction steps. We follow the benchmark evaluation protocol and report *avg@4* together with *pass@4*.

AppWorld. AppWorld (Trivedi et al., 2024) evaluates agents on tasks which interacts with APIs exposed by simulated applications. An agent must determine which applications and APIs are needed, execute sequences of calls, and use the returned information to complete the request. The tasks require dependent operations across applications, making successful completion sensitive to earlier decisions. We follow the standard evaluation protocol and report *avg@4* together with *pass@4*.

C Details of the Frameworks

Our experiments cover two forms of agent interaction and therefore use two established training frameworks. For ALFWorld, WebShop, and ScienceWorld, we build on verl-agent (Feng et al., 2025) and follow the environment implementation provided by BEACON (Wang et al., 2026d). For BFCL-v4 and AppWorld, we use the environment and rollout setup released with AgentEvolver (Zhai et al., 2025) and its SeeUPO implementation (Hu et al., 2026). We keep the environment interface, interaction protocol, and evaluation procedure of the corresponding framework, while replacing the policy-optimization component when evaluating different RL methods.

Text-based interaction framework. ALFWorld, WebShop, and ScienceWorld are implemented with the interaction pipeline inherited from verl-agent. The framework separates language generation from environment execution: at each environment step, the model receives the task context and the latest environment observation, generates one complete action, and then passes this action to the environment. The next observation is returned only after the action has been executed. This interface matches the environment-step formulation used by **SCALE**, while the policy itself remains autoregressive at the token level. We use the environment implementations from the BEACON codebase for all three benchmarks. BEACON extends `verl-agent` with support for ScienceWorld while retaining the same interaction structure for ALFWorld and WebShop. The environment manager also records information produced at each transition, such as observations, executed actions, rewards, and episode termination. We use these trajectories to construct the return and success targets required by the two critic heads. No additional intermediate annotation is introduced by **SCALE**.

Interaction context. For the three text-based environments, the model input is constructed at every environment step rather than by treating the entire episode as one generated sequence. The input contains the task description, the current observation, a limited record of previous observations

and actions, and the actions that are valid in the current state when the environment provides them. This follows the prompt and memory interface of verl-agent. The resulting structure keeps the correspondence between one model response and one environment action explicit.

The three environments use the same general interaction pattern, while adapting the prompt to the task and action space.

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 is 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 have finished your reasoning, choose an admissible action for the current step and present it within <action>...</action> tags.

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 is your turn to take one action for the current step. You should first reason step-by-step about the current situation and decide which admissible action best advances the shopping goal. This reasoning process MUST be enclosed within <think>...</think> tags. Once you have finished your reasoning, choose an admissible action and present it within <action>...</action> tags.

Prompt Template for ScienceWorld

You are an expert autonomous agent operating in the ScienceWorld environment, a text-based environment centered on tasks from the elementary science curriculum.
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 is your turn to take one action for the current step. You should first reason step-by-step about the current situation and decide which admissible action best advances the task goal. This reasoning process MUST be enclosed within <think>...</think> tags. Once you have finished your reasoning, choose an admissible action and present it within <action>...</action> tags.

The environment-specific templates differ mainly in the description of the task. ALFWorld presents household manipulation commands (Shridhar et al., 2021), WebShop presents the shopping goal and currently available page actions (Yao et al., 2022), and ScienceWorld presents the scientific task together with the actions available in the current environment state (Wang et al., 2022). We retain these prompt conventions across all compared RL methods.

Tool-use interaction framework. For BFCL-v4 and AppWorld, we follow the training setup in the SeeUPO branch of AgentEvolver (Hu et al., 2026). In this framework, the RL trainer and the benchmark environment communicate through an environment service. The trainer sends the current interaction state and the generated action to this service, which executes the requested function or

code and returns the resulting observation. This separation allows the same policy-training pipeline to interact with both benchmarks without embedding their execution logic in the optimizer.

Following the reference setting, BFCL-v4 and AppWorld use asynchronous vLLM rollouts with at most 10 interaction steps, eight rollouts per prompt, and temperature 0.9. We retain the framework's linear think context template and keep the rollout protocol fixed across backbones.

For BFCL-v4 (Patil et al., 2025), we use the multi-turn data and environment implementation in AgentEvolver. At each turn, the model receives the user request and currently available function definitions, generates a function call, and uses the returned execution result in the next turn. The implementation follows the Qwen function-call format shown below.

Prompt Template for BFCL-v4

 # Tools
You may call one or more functions to assist with the user query.
You are provided with function signatures within <tools>...</tools> tags:
<tools>
{function definitions}
</tools>

For each function call, return a JSON object containing the function name and arguments within <tool_call>...</tool_call> tags:
<tool_call>
{"name": {function name}, "arguments": {arguments}}
</tool_call>

where the available functions may change across turns. We preserve the tool definitions and execution results produced by BFCL rather than adding task-specific instructions for **SCALE**. Evaluation uses the BFCL evaluation routines provided by the framework.

For AppWorld (Trivedi et al., 2024), we use the official environment and evaluator wrapped by AgentEvolver. The model can inspect available applications and API documentation, execute Python code through the provided APIs, and use each execution result in subsequent steps. The interaction follows the prompt structure below.

Prompt Template for AppWorld

You are an AI assistant whose task is to complete the user's request by interacting with applications through their APIs.
You will undertake a multi-step interaction using a Python execution environment. At each step, write Python code, observe the returned output, and use that information to decide the next action.

To inspect the available applications:
```
print(apis.api_docs.show_app_descriptions())
```
To inspect the APIs provided by an application:
```
print(apis.api_docs.show_api_descriptions(
  app_name={app name}))
```
To inspect the specification of a particular API:
```
print(apis.api_docs.show_api_doc(
  app_name={app name},
  api_name={api name}))
```
Each code execution produces an output that can be used in subsequent steps.
Task: {user request}

This interaction continues until the task is completed or the maximum number of environment steps is reached. We use the task evaluator supplied by AppWorld to determine completion and obtain the final outcome.

Evaluation protocol. We follow the evaluation procedure provided by the corresponding benchmark framework. Table 5 summarizes the task score, success criterion, and metric used for each benchmark.

Table 5: **Evaluation protocol across the five benchmarks.** We summarize the task reward or score, the criterion for a successful trajectory, and the reported evaluation metrics.

Benchmark	Task Reward / Score	Success Criterion	Evaluation Metric
ALFWorld	Environment task completion	`info["won"] = True`	Success Rate
WebShop	$q_i \in [0, 1]$	Terminal and $q_i = 1$	Task Score / Success Rate
ScienceWorld	Final score z_i	Terminal and $z_i > 0$	Task Score / Success Rate
AppWorld	$r_i = \frac{P_i}{P_i + F_i}$	$r_i = 1$	Avg@4 / Pass@4
BFCL-v4 MT	Binary correctness $(0/1)$	Official evaluator returns correct	Avg@4 / Pass@4

For ALFWorld, WebShop, and ScienceWorld, each task is evaluated once using the environment interface inherited from verl-agent. Let $s_i^{(b)} \in \{0, 1\}$ indicate whether task i is successful on benchmark b. The success rate is

$$\mathrm{SR}^{(b)} = \frac{1}{N_b} \sum_{i=1}^{N_b} s_i^{(b)}. \tag{61}$$

For ALFWorld, success is given directly by the environment completion flag. For WebShop, a trajectory is successful only when the episode terminates with task reward 1, while the task score is the average continuous environment reward in $[0, 1]$. For ScienceWorld, we follow the legacy BEACON(Wang et al., 2026d) criterion, where `won = isCompleted and score > 0`. Since `isCompleted` may also be triggered by the environment step limit, *Succ* is a loose benchmark-specific success metric rather than strict full-task completion. This criterion is applied identically to all methods and baselines, and the Success Critic on ScienceWorld predicts this benchmark-defined binary outcome.

For BFCL-v4 and AppWorld, we follow the SeeUPO setup and sample $K = 4$ rollouts for each task. Let $r_{i,j}^{(b)} \in [0, 1]$ denote the reward of rollout j for task i. We define

$$\mathrm{Avg@}K = \frac{1}{N_b K} \sum_{i=1}^{N_b} \sum_{j=1}^{K} r_{i,j}^{(b)}. \tag{62}$$

A rollout is counted as fully successful when $r_{i,j}^{(b)} = 1$. In AppWorld, the reward is the fraction of passed goal checks, so partial completion contributes to *avg@4* but does not count as full success. In BFCL-v4 Multi-Turn Base, the reward is binary correctness, so *avg@4* is equivalent to the success rate over all sampled rollouts. We additionally define

$$\mathrm{Pass@}K = \frac{1}{N_b} \sum_{i=1}^{N_b} \mathbf{1}\left[\exists j,\ r_{i,j}^{(b)} = 1\right], \tag{63}$$

which measures the fraction of tasks solved at least once among the K sampled rollouts. Unless otherwise stated, the main results report success rate for ALFWorld, task score and success rate for WebShop and ScienceWorld, and *avg@4* for BFCL-v4 and AppWorld.

Consistency across methods. Within each benchmark and backbone, all RL methods use the same environment implementation, prompt format, rollout interface, and evaluation procedure. The differences between methods are restricted to their learning objectives and the quantities required by those objectives. In particular, **SCALE** does not introduce additional environment feedback or privileged intermediate supervision. Its Success Critic is trained from the same terminal task outcome that is available in the collected trajectories.

D Training Procedure of SCALE

This section summarizes the training procedure of **SCALE**. At each iteration, the current policy collects trajectories, the dual-head critic learns return and success prediction from the same rollout data, and the resulting return- and success-based advantages are combined according to Success Critic reliability before the standard token-level PPO update. Algorithm 1 gives the complete procedure.

Algorithm 1 Training procedure of **SCALE**.

Input: Policy π_θ, return critic V_ψ, Success Critic P_ϕ
Input: $\gamma, \lambda, \lambda_S, \beta$
for training iteration $k = 1, 2, \ldots$ **do**
  Collect trajectories $\mathcal{B}_k$ using π_θ
  Obtain return targets $\hat{G}_t$ and task outcomes S
  Cache pre-update predictions $v_t^{\text{pre}} = V_\psi(h_t), \quad p_t^{\text{pre}} = P_\phi(h_t)$ for all prefixes in $\mathcal{B}_k$
  Compute BS_k and EV_k^S from $\{p_t^{\text{pre}}\}$ on $\mathcal{B}_k$
  Compute reliability weight $c_k = q_k^{\text{BS}} q_k^{\text{EV}}$
  for each trajectory $\tau \in \mathcal{B}_k$ **do**
    Compute return advantages $\hat{A}_t^V$ from $\{v_t^{\text{pre}}\}$ using Eq. equation 1
    Set $p_t = p_t^{\text{pre}}$ and anchor $p_T = S$
    Compute $\delta_t^S = p_{t+1} - p_t$
    Compute Success-GAE $\hat{A}_t^S$ using Eq. equation 3
    Combine credits: $\hat{A}_t = \hat{A}_t^V + \beta c_k \hat{A}_t^S$
    Assign $\hat{A}_t$ to all tokens in action a_t
  end for
  Update V_ψ and P_ϕ using Eq. equation 2
  Update π_θ using Eq. equation 9
end for

Thus, **SCALE** leaves the policy objective unchanged and modifies only how environment-step credit is estimated, calibrated, and integrated before the PPO update. This preserves the standard token-level optimization procedure while allowing the actor to learn from credit signals that more directly reflect eventual task success.

E CASE STUDIES OF SCALE

To complement the quantitative results in the main paper, we present two successful trajectories from ScienceWorld and BFCL-v4. These examples show why credit assignment is difficult in long-horizon tasks. An action may have a simple immediate effect, while its importance becomes clear only after later interactions. In ScienceWorld, the agent must complete several task requirements in the correct order. In BFCL-v4, actions in early turns change the filesystem used in later turns. We report the environment observations, actions, scores, user requests, and tool outputs.

E.1 SCIENCEWORLD: ORDERED SUBGOALS AND DELAYED TASK COMPLETION

We first show a successful *find-living-thing* trajectory from ScienceWorld (Wang et al., 2022). The task contains several ordered requirements. The agent must first focus on a living thing and then move it to the orange box in the kitchen. As a result, the value of an early action depends on what the agent does in later steps. The trajectory below reports the actions, observations, and native ScienceWorld scores.

Environment (Initial State)

Task:
Find a living thing. First, focus on the thing. Then, move it to the orange box in the kitchen.

Agent (Step 1)

<think> I first need to identify a living thing in the current environment. I do not yet know what objects are available or where the target might be. The task also requires

me to focus on the object before moving it, so I should first inspect the surroundings and then choose a suitable target. </think>

<action> look around </action>

Environment (Step 1)

Score: 0.08.

This location is called the outside. You see a blue jay egg, a butterfly egg, a dove egg, an axe, and a door to the kitchen.

Agent (Step 2)

<think> I can now see several eggs, which are possible living things for this task. The blue jay egg is a clear candidate, so I will use it as the target. Since the task explicitly requires focusing on the object before moving it, I should complete that requirement first. </think>

<action> focus on blue jay </action>

Environment (Step 2)

Score: 0.58.

You focus on the blue jay egg.

Agent (Step 3)

<think> The blue jay has now been selected and focused on, so the first requirement is satisfied. The final goal is to move it to the orange box in the kitchen. Before I can travel to the destination, I need to take the object with me, so picking it up is the next necessary step. </think>

<action> pick up blue jay </action>

Environment (Step 3)

Score: 0.67.

You move the blue jay to the inventory.

Agent (Step 4)

<think> The blue jay is now in my inventory, which means I can carry it to the required destination. The task says that the orange box is located in the kitchen, while I am currently outside. I should therefore move to the kitchen and look for the orange box there. </think>

<action> go to kitchen </action>

Environment (Step 4)

Score: 0.83.

You move to the kitchen. You see a counter, a table, and an orange box.

Agent (Step 5)

<think> I have reached the kitchen and can see the orange box. I am already carrying the blue jay, and the earlier focusing requirement has been completed. The only remaining part of the task is to place the target into the orange box, which should complete the task. </think>

<action> move blue jay to orange box </action>

Environment (Outcome)

Score: 1.00.

You move the blue jay to the orange box.

Task Completed.

Credit-assignment perspective. The focus action is important because it satisfies a required part of the task before the agent moves the object. Its effect cannot be judged from that step alone. The agent must still pick up the blue jay, move to the kitchen, and place it in the orange box. This example shows why an intermediate action should be evaluated by how it affects the chance of completing the full task. In SCALE, the Success Value estimates this probability from each interaction prefix, and changes in the Success Value provide the signal used by Success-GAE.

E.2 BFCL-v4: Cross-Turn Credit under Persistent Tool State

We next show the official BFCL-v4 Multi-Turn Base instance `multi_turn_base_0` (Patil et al., 2025), using the ground-truth interaction driver in AgentEvolver. The task contains four user turns, and the filesystem is shared across them. Actions in one turn can therefore affect what is possible in later turns. For example, moving a file in the first turn changes the location used by later search, sorting, and comparison operations. Multiple filesystem calls may be made within each turn.

Environment (Initial State)

The initial `GorillaFileSystem` contains

`workspace/document/final_report.pdf`
with content

```
Year2024 This is the final report content
including budget analysis and other
sections.
```

and
`workspace/document/previous_report.pdf`
with content

```
Year203 This is the previous report content
with different budget analysis.
```

Environment (Turn 1)

User request:

Move 'final_report.pdf' within document directory to 'temp' directory in document. Make sure to create the directory.

Agent (Turn 1)

```
<tool_call>
{"name": "cd", "arguments": {"folder": "document"}}
</tool_call>
<tool_call>
{"name": "mkdir", "arguments": {"dir_name": "temp"}}
</tool_call>
<tool_call>
{"name": "mv", "arguments": {"source":
"final_report.pdf",
"destination": "temp"}}
</tool_call>
```

Environment (Turn 1: Tool Results)

```
<tool_call>
{"name": "cd_1", "content":
{"current_working_directory": "document"}}
</tool_call>
<tool_call>
{"name": "mkdir_2", "content": "None"}
</tool_call>
<tool_call>
{"name": "mv_3", "content":
{"result": "'final_report.pdf' moved to
'temp/final_report.pdf'"}}
</tool_call>
```

Agent (Turn 1: Advance)

```
ok.1
```

Environment (Turn 2)

User request:
Perform a detailed search using grep to identify sections in the file pertaining to 'budget analysis'.

Agent (Turn 2)

```
<tool_call>
{"name": "cd", "arguments": {"folder": "temp"}}
</tool_call>
<tool_call>
{"name": "grep", "arguments":
{"file_name": "final_report.pdf",
"pattern": "budget analysis"}}
</tool_call>
```

Environment (Turn 2: Tool Results)

```
<tool_call>
{"name": "cd_1", "content":
{"current_working_directory": "temp"}}
</tool_call>
<tool_call>
{"name": "grep_2", "content":
{"matching_lines": [
"Year2024 This is the final report content including
budget analysis and other sections."
]}}
</tool_call>
```

Agent (Turn 2: Advance)

```
ok.2
```

Environment (Turn 3)

User request:
Upon identifying the requisite `'budget analysis'` content, sort the `'final_report.pdf'` by line for improved clarity and comprehension.

Agent (Turn 3)

```
<tool_call>
{"name": "sort", "arguments":
{"file_name": "final_report.pdf"}}
</tool_call>
```

Environment (Turn 3: Tool Result)

```
<tool_call>
{"name": "sort_1", "content":
{"sorted_content":
"Year2024 This is the final report content including
budget analysis and other sections."}}
</tool_call>
```

The BFCL filesystem implementation returns the sorted content; the `sort` operation does not overwrite the file.

Agent (Turn 3: Advance)

```
ok.2
```

Environment (Turn 4)

User request:
Move `'previous_report.pdf'` in document directory to temp as well and having final report also there, proceed to juxtapose it with `'previous_report.pdf'` to detect any critical alterations.

Agent (Turn 4)

```
<tool_call>
{"name": "cd", "arguments": {"folder": ".."}}
</tool_call>

<tool_call>
{"name": "mv", "arguments":
{"source": "previous_report.pdf",
"destination": "temp"}}
</tool_call>

<tool_call>
{"name": "cd", "arguments": {"folder": "temp"}}
</tool_call>

<tool_call>
{"name": "diff", "arguments":
{"file_name1": "final_report.pdf",
"file_name2": "previous_report.pdf"}}
</tool_call>
```

Environment (Turn 4: Tool Results)

```
<tool_call>
{"name": "cd_1", "content": {}}
</tool_call>

<tool_call>
{"name": "mv_2", "content":
{"result": "'previous_report.pdf' moved to
'temp/previous_report.pdf'"}}
</tool_call>

<tool_call>
{"name": "cd_3", "content":
{"current_working_directory": "temp"}}
</tool_call>

<tool_call>
{"name": "diff_4", "content":
{"diff_lines":
"- Year2024 This is the final report content including
budget analysis and other sections.
+ Year203 This is the previous report content with
different budget analysis."}}
</tool_call>
```

Agent (Turn 4: Advance)

```
ok.2
```

Environment (Outcome)

```
[CONVERSATION_COMPLETED]
```

The four per-turn function-call groups correspond to the reference ground-truth interaction for `multi_turn_base_0`. AgentEvolver subsequently invokes the BFCL sparse evaluator on the completed conversation.

Credit-assignment perspective. The `mkdir` and `mv` calls in the first turn prepare the filesystem for the later requests. After `final_report.pdf` is moved to `temp`, later turns access the same file with `grep` and `sort`. The file is also used in the final `diff` operation. The importance of the first-turn actions therefore becomes clear only after several later turns. This example shows that credit for an early tool call may depend on whether the state it creates supports successful completion of later requests.

Discussion. The two examples show two common sources of delayed credit. In ScienceWorld, an early action satisfies one requirement, but the task succeeds only after several later actions. In BFCL-v4, an early tool call changes the filesystem used in later turns. In both cases, the effect of an action cannot be judged only from its immediate result. It also depends on whether the resulting interaction prefix makes eventual task completion more likely. This is the motivation for using Success Value in SCALE and for using its changes to construct Success-GAE.