Verlog: Context-lite Multi-turn Reinforcement Learning framework for Long-Horizon LLM Agents

Wen-Tse Chen
Carnegie Mellon University
wentsec@cs.cmu.edu

Jiayu Chen
The University of Hong Kong
jiayuc@hku.hk

Hao Zhu
Stanford University
zhuhao@stanford.edu

Jeff Schneider
Carnegie Mellon University
jeff4@cs.cmu.edu

Abstract

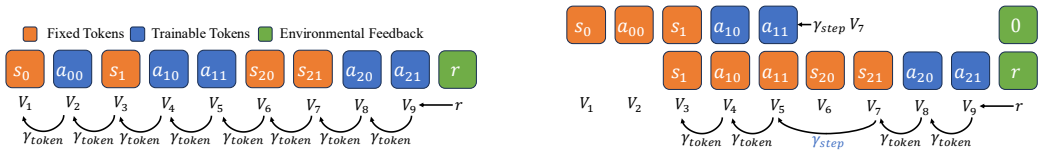
Finetuning large language model (LLM) agents with multi-turn reinforcement learning (RL) is a promising direction. However, applying multi-turn RL to long-horizon agentic tasks presents unique challenges not typically encountered in reasoning tasks such as solving math problems. These include long interaction histories that hinder relevant context retrieval, sparse rewards that slow down learning, and variable trajectory lengths that reduce training efficiency. To address these challenges, we propose Verlog, a framework that incorporates: (1) customizable agent memory mechanism, allowing the agent to flexibly include different lengths of historical interaction in each turn's prompt based on task requirements, and (2) Dual-discounting GAE, which decouples step-level and token-level credit assignment. (3) trajectory early truncation, which reduces GPU idle time and boosts multi-turn RL training efficiency. Experiments demonstrate that our method surpasses the zero-shot performance of state-of-the-art LLMs across three benchmarks, BabyAI, BabaIsAI and Crafter, while also achieving greater efficiency and effectiveness than variants lacking either the memory mechanism or dual-discounting GAE. Notably, Verlog is the first framework capable of training LLM agents on trajectories exceeding 400 turns, demonstrating scalability far beyond prior approaches. Our code is available on GitHub.

1 Introduction

Reinforcement learning (RL) has been widely applied to reasoning tasks to enhance the deep thinking capabilities of large language models (LLMs) Guo et al. [2025], Pan et al. [2025], and recent work has extended RL to multi-turn settings with promising results Zhou et al. [2025], Chen et al. [2024]. However, multi-turn tasks differ significantly from typical reasoning tasks, posing challenges for directly applying existing RL methods. First, during inference, as the number of turns increases, LLM agents struggle to extract task-relevant information from overly long histories Laban et al. [2025]. Second, during training, longer trajectories result in sparser rewards, since reward signals are typically provided only at the end, thereby hindering effective learning. At the system level, large variance in trajectory lengths results in inefficient GPU utilization, as shorter trajectories must wait for longer ones to finish.

To address these issues, we propose Verlog, a context-lite multi-turn RL framework, which has the following advantages: (1) It supports customizable agent memory mechanism, allowing users to design agent memory mechanisms tailored to specific tasks rather than always using the entire

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Multi-Turn Interactions in Large Language Models.



(a) Multi-turn PPO with full memory.

(b) Multi-turn PPO with memory length 1.

Figure 1: (a) Prior work applies PPO at the token level, where each action token a_{ti} is generated based on all preceding information: $s_0, a_0, \cdots, s_t, a_{t0:t(i-1)}$. (b) In contrast, our algorithm limits the context length and introduces a dual discounting strategy for PPO training. Specifically, when computing GAEs, we apply $\gamma_{\text{token}}, \lambda_{\text{token}}$ within individual turns and $\gamma_{\text{step}}, \lambda_{\text{step}}$ across turns, as illustrated by the arrows. Although the first turn does not receive a reward, we can still leverage $\gamma_{\text{step}} V_7$ as a training signal.

trajectory as input, which we show improves training efficiency and convergent performance in agentic tasks such as BabyAI. (2) It adopts dual discounting GAE for finer-grained credit assignment. Specifically, a larger discount factor is applied to tokens within a turn to encourage extended reasoning, while a smaller discount factor is used across turns to discourage unnecessarily long dialogues. (3) It enables fixed batch training with trajectories early truncation, significantly improving GPU utilization.

2 Related Works

We compare our method against existing multi-turn RL frameworks for training LLM agents. RAGEN Wang et al. [2025] supports multi-turn RL but is limited to tasks with short decision horizons (5-10 turns). VeRL Sheng et al. [2024] enables asynchronous rollouts, improving efficiency when response lengths vary across turns, but does not address challenges posed by a large and variable number of dialogue turns. SkyRL Cao et al. [2025] supports long-horizon tasks and asynchronous environments, but does not explore efficient memory mechanisms for multi-turn RL. In contrast, our method supports long-horizon, multi-turn tasks, enabling effective credit assignment by using different discount factors (and thus different effective horizons) at the token and step levels.

We also notice KIMI K1.5 Team et al. [2025], a single-turn RL method that handles over-length responses by truncating and storing them in the replay buffer, continuing generation in subsequent training steps. However, this approach is incompatible with PPO-based multi-turn extensions, as PPO is an on-policy algorithm that requires responses to be sampled from the current policy.

3 Preliminary

A Markov Decision Process (MDP) can be described as a tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho_0)$. Here, $\mathcal{S}$ and $\mathcal{A}$ represent the state and action space, respectively; $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition kernel; $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function; $\gamma \in [0, 1)$ is the discount factor. The goal of RL agents is to learn a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ that maximizes the expected return:

$$\mathbb{E}_{s_0, a_0, s_1, \cdots} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right], \tag{1}$$

where $s_0 \sim \rho_0(\cdot), a_t \sim \pi(\cdot|s_t), s_{t+1} \sim P(\cdot|s_t, a_t)$. LLM agent tasks involve multiple turns. At each turn t, the agent receives a state prompt s_t (task description + feedback from the previous turn) and outputs a series of action tokens a_t (e.g., tool calling or acting). Agents are fine-tuned via multi-turn RL with verifiable, often sparse rewards to improve sequential decision-making.

4 Methodology

In the following section, we address two key challenges in multi-turn RL training for LLM agents: designing efficient agent memory mechanisms and assigning temporal credit across dialogue turns.

4.1 Context-lite Multi-turn RL

Table 1: Win Rate (%) across BabyAI tasks for different memory length. The results show that shorter memory lengths (1 to 4) generally lead to higher performance across BabyAI tasks. Values are mean ± standard error.

Memory length	**1**	**2**	**4**	**8**	**16**	**32**
BabyAI (avg)	31.87±3.68	28.13±3.55	25.00±3.42	17.50±3.00	14.47±2.79	15.72±2.89
goto	87.50±5.85	81.25±6.90	50.00±8.84	53.13±8.82	46.88±8.82	37.50±8.56
pickup	40.63±8.68	28.13±7.95	25.00±7.65	18.75±6.90	15.63±6.42	28.13±7.95
pick→goto	21.88±7.31	21.88±7.31	34.38±8.40	6.25±4.28	6.25±4.28	9.68±5.31
open	9.38±5.15	9.38±5.15	15.63±6.42	6.25±4.28	3.13±3.08	3.13±3.08

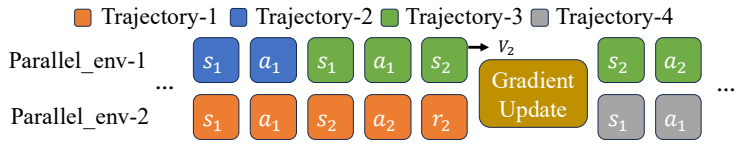


Figure 2: Each trajectory is shown in a different color. For each gradient update, a fixed number of turns are executed. If a trajectory does not terminate within the rollout, the value function of the final state is used as a training signal to guide the LLM agent.

Prior work Wang et al. [2025] typically treats the entire trajectory (i.e., $\tau = [s_0, a_0, \cdots, s_T, a_T]$) as a single training data point, with the full trajectory as the context, a reward signal at the end, and training only the action tokens (as shown in Figure 1(a)). In particular, the policy is defined and trained as $\pi(a_0, \cdots, a_T|\tau)$, where the historical information $s_0, a_0, \cdots, s_t$ is treated as valid input when generating each action a_t.

Using such long contexts can lead to inefficient RL training, as it imposes high memory demands and may cause the LLM to lose focus on decision-making at the current time step. In contrast, our framework enables more customizable and granular context usage. Specifically, we treat each turn as an individual training data point, allowing flexible control over how many previous turns are included in the current prompt s_t.

As illustrated in Figure 1(b), we truncate outdated state-action pairs from the trajectory and retain only the most recent *memory length* state-action pairs along with the current state in the context window. The second row in Figure 1(b) demonstrates the case where the memory length is set to one and the data point at each turn t involves (s_{t-1}, a_{t-1}, s_t) as the context and a_t as the action.

Early Trajectory Truncation in PPO Training: When the number of turns in a trajectory exceeds the training batch size, the reward signal may not be immediately available. To address this, our PPO implementation supports early truncation of trajectories, using the value of the final state as a training signal. It is very common for the number of turns in a trajectory to exceed the training batch size. For example, consider a training batch size of 256. To improve inference efficiency, practitioners often increase the number of parallel environments since rollout time becomes the bottleneck in multi-turn RL training. If 16 parallel environments are used, then any task with an episode length exceeding 16 turns may lead to issues in prior frameworks that lack support for early truncation. As shown in Figure 2, this design offers an additional benefit: when trajectory lengths vary significantly, we can truncate trajectories as soon as enough turns have been collected, without waiting for the longest rollout to complete. This improves the overall system throughput.

4.2 Dual Discounting Strategy for Multi-turn RL

In single-turn RL fine-tuning, we typically want to avoid response length shrinkage after training. A common practice is to set the token-level discount factor, γ_{token}, to 1. However, in multi-turn RL fine-tuning, our goal often shifts toward encouraging the agent to complete the task efficiently, minimizing the number of dialogue turns, which can be achieved by using a step-level discount factor $\gamma_{\text{step}} < 1$. Unfortunately, this creates a tension with the need for longer, more coherent reasoning paths, which require more tokens per turn. To address this conflict, we propose a **dual-discounting strategy** for multi-turn RL Generalized Advantage Estimates (GAE) Schulman et al. [2015] approximation, where we decouple the token-level discount factors $(\gamma_{\text{token}}, \lambda_{\text{token}})$ from the step-level discount factors $(\gamma_{\text{step}}, \lambda_{\text{step}})$, when computing GAE. This approach allows us to independently control reasoning granularity within a step and the overall conversational efficiency across steps. We set $\gamma_{\text{step}} = 0.99, \lambda_{\text{step}} = 0.95, \gamma_{\text{token}} = 1, \lambda_{\text{token}} = 1$ in this work.

Table 2: Crafter results using Qwen2.5-7B-Instruct model with PPO.

Metric	Instruct-model	Verlog (Ours)
Rewards	5.80	**10.44**
Avg. Trajectory Length	172.23	**196.42**

Table 3: BabaIsAI and BabyAI win rates using Qwen2.5-3B-Instruct model with PPO. Values are mean ± standard error.

Env	Task	Instruct-model	Verlog (Ours)
BabaIsAI	goto+distr_obj	0.66 ± 0.08	**1.00** ± **0.00**
	two_room+goto	0.03 ± 0.03	**0.89** ± **0.17**
	two_room+goto+distr_rule	0.22 ± 0.07	**0.89** ± **0.11**
	two_room+goto+maybe_break	0.19 ± 0.07	**0.36** ± **0.07**
BabyAI	goto	0.88 ± 0.06	**1.00** ± **0.00**
	pickup	0.41 ± 0.09	**1.00** ± **0.00**
	pickup->goto	0.22 ± 0.07	**0.65** ± **0.16**
	open	0.09 ± 0.05	**0.94** ± **0.07**

With the dual discounting strategy, the GAE formulation is recursively defined as follows:

$$\hat{A}_t = \gamma\lambda\hat{A}_{t+1} + \delta_t^V, \tag{2}$$

where $\gamma\lambda = \gamma_{\text{step}}\lambda_{\text{step}}$ if token t and token $t+1$ are in the different turns and $\gamma\lambda = \gamma_{\text{token}}\lambda_{\text{token}}$ otherwise. δ_t^V, i.e., the TD-residual, is defined as $\delta_t^V = -V(s_t) + r_t + \gamma V(s_{t+1})$, where $V(s_t)$ is the value function. Note that both states and actions consist of multiple tokens. However, the recursive process described in Equation (2) is not applied between state tokens, as states are not generated by the LLM and can be treated as a single chunk of input.

5 Results

In this section, we address the following research questions (RQs): **RQ1**: What is the impact of the memory length on the performance of LLM agents in multi-turn tasks? **RQ2**: Can our proposed multi-turn RL fine-tuning approach improve the performance of LLM agents compared to their zero-shot capabilities? **RQ3**: Does the proposed dual discounting strategy improve value function approximation and lead to improved performance of LLM agents? **RQ4**: How would the memory length impact multi-turn RL fine-tuning for LLM agents? We evaluate our algorithms on BabyAI Carta et al. [2023], BabaIsAI Cloos et al. [2024] and Crafter Hafner [2021], each with a maximum episode length ranging from 100 to 2000 steps. In all settings, both the policy inputs (observations) and outputs (actions) are represented in text form. The action space is discrete and enumerable.

5.1 Impact of Memory Length on Zero-Shot Capabilities of LLM Agents (RQ1)

We evaluate the zero-shot performance of Qwen-2.5-3B-Instruct in the BabyAI environment using different memory lengths. We define the memory length as the number of previous turns included in the policy's context window. Unlike prior work that defaults to including nearly the entire trajectory history (with memory length fixed at 32 for BabyAI tasks), our implementation enables flexible memory configurations tailored to specific tasks.

As shown in Table 1, this flexibility yields substantial gains: with proper memory length, performance improves by over **2×** compared to the baseline. Interestingly, we observe that simpler tasks, such as *goto* and *pickup*, perform best with memory length 1, while more complex tasks like *open* and *pick_up_seq_go_to* benefit most from memory length 4.

This simple memory mechanism already demonstrates significant potential, highlighting that memory design is a critical yet underexplored component of LLM agents.

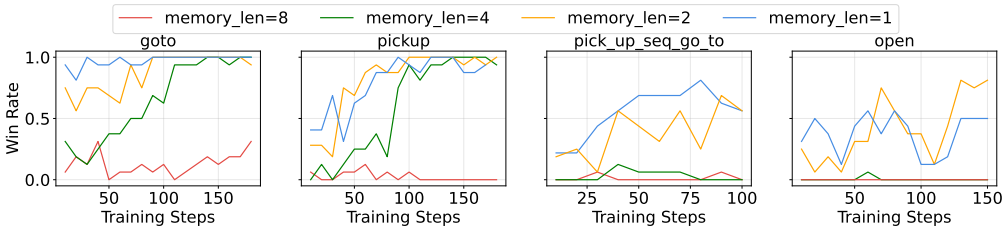


Figure 4: Effect of memory length on multi-turn RL fine-tuning. Shorter memory lengths (1–2) lead to higher performance, likely due to improved zero-shot behavior and denser reward signals. Incorporating limited context supports consistent reasoning while avoiding the inefficiencies introduced by longer histories.

5.2 Benchmarking Verlog (RQ2)

In this subsection, we evaluate the performance of our proposed method on three distinct benchmarks, BabyAI, BabaIsAI, and Crafter, and compare it against the zero-shot performance of the Qwen-2.5-Instruct model Yang et al. [2025]. The results are summarized in Table 2 and Table 3. For BabyAI and BabaIsAI, we use the Qwen2.5-3B-Instruct model fine-tuned with the PPO algorithm. Training is conducted on 4×A40 GPUs (48 GB each) for approximately 24 hours, corresponding to 300 PPO updates. The maximum episode lengths are set to 128 and 100, respectively. We report the mean win rate across three runs with different random seeds. For Crafter, we use the larger Qwen2.5-7B-Instruct model with PPO, trained on 8×H100 GPUs (82 GB each) for approximately 36 hours, corresponding to 170 PPO updates.

Across all benchmarks, fine-tuning significantly improves both win rates and task scores compared to zero-shot performance. These results demonstrate that Verlog enables RL policies to effectively learn from long-horizon interactions with the environment.PPO updates. Training curve can be found in Figure 5.

5.3 Verlog Benefits from Dual Discounting GAE (RQ3)

We conduct an ablation study on the proposed dual discounting GAE and demonstrate its effectiveness in improving value function estimation and enhancing the sample efficiency of multi-turn RL training for long-horizon LLM agents. As shown in Figure 3, agents trained with dual discounting outperform those using the baseline configuration ($\gamma_{\text{step}} = \lambda_{\text{step}} = \gamma_{\text{token}} = \lambda_{\text{token}} = 1$) in the BabyAI *pickup* task, exhibiting higher sample efficiency. This improvement is primarily due to the step-level discounting mechanism, which enables more effective temporal credit assignment. Furthermore, dual discounting GAE leads to lower value prediction errors, reflecting a more stable and reliable training process.

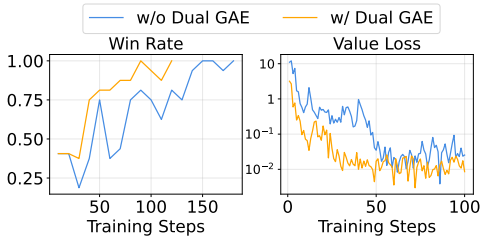


Figure 3: LLM agents trained with dual discounting GAE show faster convergence, and lower PPO value loss on the challenging BabyAI *pickup* scenario.

5.4 Impact of Memory Length on Verlog (RQ4)

In this subsection, we investigate the effect of memory length on multi-turn RL fine-tuning for long-horizon LLM agents. As shown in Table 4, the LLM agent achieves higher performance when using a shorter memory length during fine-tuning. In particular, memory lengths of one and two consistently yield the highest performance across settings. We observe that longer memory lengths can lead to lower zero-shot performance, which results in sparser reward signals and less efficient RL fine-tuning. Additionally, the agent's reasoning paths often reference or revise plans from previous turns. This behavior appears to enhance planning consistency across turns, suggesting that RL fine-tuning with contextual information is more effective than fine-tuning without context.

6 Conclusion

In this work, we propose Verlog, a framework designed to address key challenges in fine-tuning LLM agents for multi-turn tasks. By introducing customizable memory length and a dual-discounting GAE, our approach tackles issues of long interaction histories, sparse reward signals, and inefficiencies arising from variable-length trajectories. Through extensive experiments on BabyAI, BabaIsAI and Crafter, we systematically investigate the impact of memory design and discounting strategies on LLM performance and demonstrate the state-of-the-art performance of our algorithm in long-horizon multi-turn agentic tasks. A limitation of our approach is that it does not support value-function-free RL fine-tuning methods, such as GRPO Shao et al. [2024] and RLOO Ahmadian et al. [2024].

7 Acknowledgement

This work was supported in part by the U.S. Army Futures Command under Contract No. W519TC-23-C-0030.

This work used Delta at the National Center for Supercomputing Applications (NCSA) through allocation CIS250651 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program [Boerner et al., 2023], which is supported by U.S. National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

References

A. Ahmadian, C. Cremer, M. Gallé, M. Fadaee, J. Kreutzer, O. Pietquin, A. Üstün, and S. Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.

T. J. Boerner, S. Deems, T. R. Furlani, S. L. Knuth, and J. Towns. Access: Advancing innovation: Nsf's advanced cyberinfrastructure coordination ecosystem: Services & support. In *Practice and experience in advanced research computing 2023: Computing for the common good*, pages 173–176. 2023.

S. Cao, S. Hegde, D. Li, T. Griggs, S. Liu, E. Tang, J. Pan, X. Wang, A. Malik, G. Neubig, K. Hakhamaneshi, R. Liaw, P. Moritz, M. Zaharia, J. E. Gonzalez, and I. Stoica. Skyrl-v0: Train real-world long-horizon agents via reinforcement learning, 2025.

T. Carta, C. Romac, T. Wolf, S. Lamprier, O. Sigaud, and P.-Y. Oudeyer. Grounding large language models in interactive environments with online reinforcement learning. In *International Conference on Machine Learning*, pages 3676–3713. PMLR, 2023.

W. Chen, J. Chen, F. Tajwar, H. Zhu, X. Duan, R. Salakhutdinov, and J. Schneider. Fine-tuning LLM agents with retrospective in-context online learning. In *Adaptive Foundation Models: Evolving AI for Personalized and Efficient Learning*, 2024.

N. Cloos, M. Jens, M. Naim, Y.-L. Kuo, I. Cases, A. Barbu, and C. J. Cueva. Baba is ai: Break the rules to beat the benchmark. *arXiv preprint arXiv:2407.13729*, 2024.

D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

D. Hafner. Benchmarking the spectrum of agent capabilities. *arXiv preprint arXiv:2109.06780*, 2021.

P. Laban, H. Hayashi, Y. Zhou, and J. Neville. Llms get lost in multi-turn conversation. *arXiv preprint arXiv:2505.06120*, 2025.

J. Pan, J. Zhang, X. Wang, L. Yuan, H. Peng, and A. Suhr. Tinyzero. https://github.com/Jiayi-Pan/TinyZero, 2025. Accessed: 2025-01-24.

J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

G. Sheng, C. Zhang, Z. Ye, X. Wu, W. Zhang, R. Zhang, Y. Peng, H. Lin, and C. Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*, 2024.

K. Team, A. Du, B. Gao, B. Xing, C. Jiang, C. Chen, C. Li, C. Xiao, C. Du, C. Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

Z. Wang, K. Wang, Q. Wang, P. Zhang, L. Li, Z. Yang, K. Yu, M. N. Nguyen, L. Liu, E. Gottlieb, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025.

A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Y. Zhou, S. Jiang, Y. Tian, J. Weston, S. Levine, S. Sukhbaatar, and X. Li. Sweet-rl: Training multi-turn llm agents on collaborative reasoning tasks. *arXiv preprint arXiv:2503.15478*, 2025.

A Training Curves

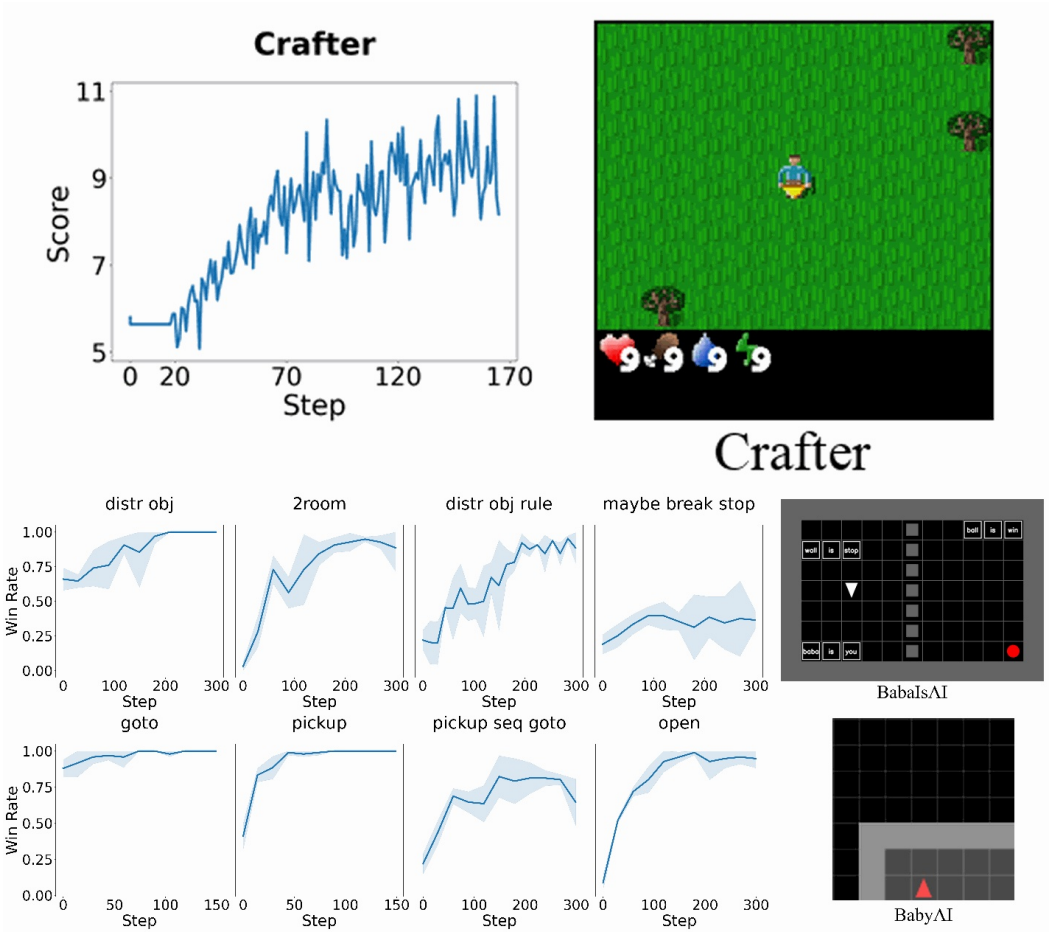


Figure 5: The curves on the left depict Verlog's training performance across nine different scenarios, with the shaded regions representing the standard error over three random seeds. The illustration on the right shows a screenshot of the environment used for policy training.

B Technical Report

In the following sections, we outline our design choices, implementation details, and explore potential research questions that our framework may help address.

B.1 Model & Prompt

B.1.1 Instruct Model

We begin with the Instruct variant of Qwen-2.5 (Qwen-2.5-3B/7B-Instruct), rather than the base model, for two key reasons. First, it enables seamless integration with BALROG, a framework designed to evaluate the zero-shot performance of instruct models across a range of benchmarks. Second, it allows us to use the benchmark's prompts with minimal modifications

B.1.2 Memory Mechanism

Rather than placing the entire trajectory into the context window, we include only the latest $n + 1$ turns. Each turn, i.e., data = $(\text{history}_t, s_t, \text{think}_t, a_t)$, with $\text{history}_t = \{s_{t-n}, \text{think}_{t-n}, a_{t-n}, ..., s_{t-1}, \text{think}_{t-1}, a_{t-1}\}$, is treated as an individual training data point. As a result, each training batch consists of `batch_size` individual turns, not `batch_size` full trajectories.

The results show that for the 3B Qwen model, performance peaks at $n = 1$ or 2 and degrades as n increases to 4 or 8. We hypothesize that this decline is due to the 3B model's limited capacity to handle long contexts—for example, $n = 8$ yields a prompt of approximately 4.6k tokens. Whether this trend holds for larger models is an open question. Notably, the tasks we evaluate can be framed as Markov Decision Processes (MDPs). In more complex or partially observable tasks, a larger n may help.

We observed two notable issues related to the multi-turn memory mechanism:

- **Mimicking prior reasoning patterns:** The model tends to replicate reasoning styles from earlier turns, reducing the diversity of its thought processes.
- **Multi-turn hallucinations:** The model struggles to distinguish between actions imagined during reasoning and actual events in the environment. For example, it may plan to "chop a tree then craft a pickaxe" but fail to find a tree in reality—yet still act as if the plan succeeded. This is a unique challenge for agentic tasks.

We conducted preliminary experiments to address these issues: (1) We tested a variant that includes only the final action in history: data = $(\text{history}_t, s_t, \text{think}_t, a_t)$, with $\text{history}_t = \{s_{t-n}, a_{t-n}, ..., s_{t-1}, a_{t-1}\}$. (2) We tested a variant that periodically clears the history buffer (every 5 steps). Both approaches led to worse performance.

B.1.3 Prompt Template

Belows is the prompt template used for BabyAI. The prompts are adapted from BALROG.

```
[SYSTEM] You are an agent playing a simple navigation game. Your goal is to
{MISSION}. The following are the possible actions you can take in the game,
followed by a short description of each action: {AVAILABLE ACTIONS}. In a
moment I will present you an observation. Tips: {TIPS}. PLAY!

[USER] {OBSERVATION}

[ASSISTANT] THINK: {THINK} ACTION: {ACTION}

[USER] {OBSERVATION}. What will you do next? Please respond in the following
format: THINK: step-by-step reasoning. ACTION: One valid action from the
allowed set.
```

We recommend always examining the model's zero-shot outputs before training. Specifically, evaluate: (1) Whether reasoning paths are diverse, (2) whether the model reasons sufficiently before selecting

an action, (3) the ratio of valid actions, and (4) the types of failure cases. These checks ensure the model understands the environment from the prompt. If not, revise the prompt before fine-tuning.

B.2 Environment

Verlog uses a highly abstract game as its testbed, reducing the need for prompt engineering and allowing researchers to focus on algorithmic design. We detail all engineering aspects below:

B.2.1 Valid Action

Improving the valid action ratio through prompt engineering is the simplest and most effective way to boost performance. In our setup, we ensure the model produces valid actions over 95% of the time using the following strategies:

- **Hardcoded action translation:** Certain invalid actions are frequently produced by zero-shot LLMs (e.g., "Move forward" and "Go forward"). We implement a hand-crafted translation function to map these to valid actions, preventing them from lowering the valid action ratio.
- **Replace invalid actions with a default action:** When the LLM outputs an invalid action, the environment rejects it and executes a predefined default action instead. Simultaneously, we replace the invalid action with the default one before appending it to the history buffer. This prevents the agent from mimicking the invalid action in subsequent steps.

We observe that truncating the trajectory upon encountering an invalid action leads to worse performance. Replacing invalid actions with a default action yields better results. In this work, we apply a 0.1 penalty to invalid actions. However, with a high valid action ratio, the format penalty has minimal impact on overall performance.

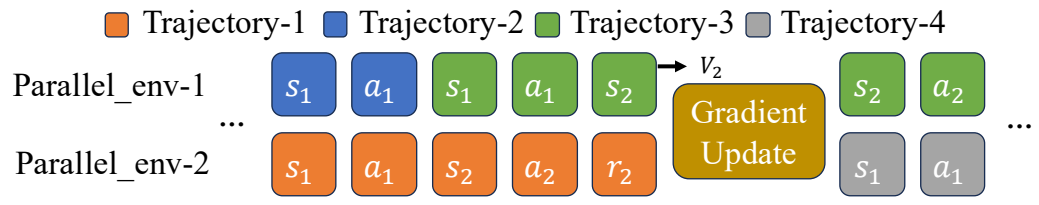
B.2.2 Reward

Rewards are rule-based and provided by the environment. In BabyAI and BabaIsAI, we adopt a binary trajectory-level reward scheme: 1 for success trajectory, 0 for failure trajectory. Combined with dual-discount GAE, this setup ensures that earlier steps in suboptimal trajectories receive lower credit compared to those in optimal ones. For Crafter, we use the native environment rewards directly.

We observed a frustrating issue when training on Crafter:

- the score improvement mainly comes from reinforcing skills that the base model already possessed, rather than learning new skills. For example, before fine-tuning, agents rarely placed furnace, but after fine-tuning, they successfully place furnace almost every episode. However, skills that were previously unknown, such as crafting an iron sword, remain unlearned even after fine-tuning. This suggests that the current version of RL fails to teach agents new skills on these tasks, instead primarily sharpening the action distribution to favor behaviors with higher immediate rewards.

B.2.3 Batch Environment (Fixed-Turn Batching)



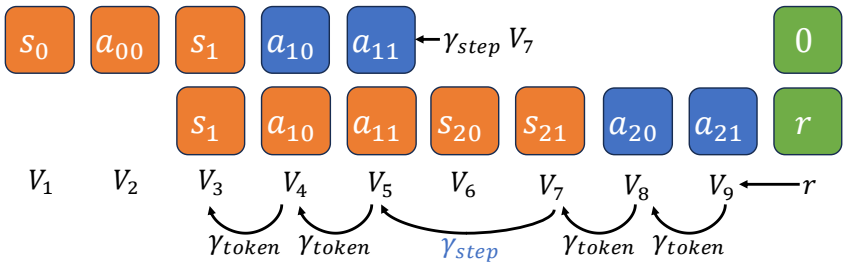
Our framework supports asynchronous rollouts and works with any environment using the OpenAI Gym interface. Each training batch size is: `n_env` × `e_len`, where:

- `n_env` = number of parallel environments
- `e_len` = episode length per rollout

Note: `e_len` can be smaller than the environment's trajectory length. For example, we set `e_len = 8` and max trajectory length = 128 in BabyAI. For early truncated trajectories, we leverage the value function to guide the training process. A longer `e_len` (smaller `n_env`) often leads to better performance, albeit at the cost of lower token throughput.

B.3 Algorithm

B.3.1 Dual Discounting GAE



To incentivize agents to solve tasks with fewer environment steps, we decouple token-level discounting $(\gamma_{\text{token}}, \lambda_{\text{token}})$ and step-level $(\gamma_{\text{step}}, \lambda_{\text{step}})$. We set:

- $\gamma_{\text{step}} = 0.99$, $\lambda_{\text{step}} = 0.95$
- $\gamma_{\text{token}} = 1.0$, $\lambda_{\text{token}} = 1.0$

The GAE is computed recursively:

$$\hat{A}_t = \gamma\lambda\hat{A}_{t+1} + \delta_t^V$$

where:

- $\gamma\lambda = \gamma_{\text{step}}\lambda_{\text{step}}$, if tokens are from different turns
- $\gamma\lambda = \gamma_{\text{token}}\lambda_{\text{token}}$, otherwise
- and $\delta_t^V = -V(s_t) + r_t + \gamma V(s_{t+1})$

The recursion starts from the last token of the final turn and proceeds backward. Once all tokens in the final turn are processed, we move to the last token of the second-to-last turn, and continue this process recursively. During this process, all state tokens are skipped. If a trajectory is truncated at step T, we store the next state s_{T+1} but do not sample a_{T+1}. Instead, we use the final token of s_{T+1} to estimate $V(s_{T+1})$, used as the bootstrap value in GAE.

B.3.2 Value Function Estimation

- When both γ and λ are set to 1.0, the value function serves purely as a baseline in PPO's advantage estimation. Specifically, the advantage for the t-th token in the last turn is defined as $A_{-1,t} = r - V_{-1,t}$, where r is the trajectory reward and $V_{-1,t}$ is the value estimate for the t-th token in the last turn.
- When λ are less than 1.0, the value function contributes to the GAE objective beyond serving as a simple baseline. For instance, in our setting with $\lambda_{\text{step}} = 0.95$, $\gamma_{\text{token}} = 1.0$, $\lambda_{\text{token}} = 1.0$, and the reward r that is zero along the trajectory except at the final turn, the advantage for the t-th token in the second-to-last turn is given by: $A_{-2,t} = \gamma_{\text{step}}[\lambda_{\text{step}} r + (1 - \lambda_{\text{step}})V_{-1,0}] - V_{-2,t}$. This indicates that, in our setting, the value function of the first token in each turn is used to bootstrap the GAE objective for the preceding turn.
- In our setting, the value of the first token of each turn carries more semantic significance than the subsequent tokens, we assign it a higher weight when training the critic network.

B.3.3 Critic Warmup

In our setting, we warm up the critic before fine-tuning, as it is used both for bootstrapping truncated trajectories and for computing GAE. That is, we freeze the actor and update only the critic at the beginning of training. Specifically, We collect `w_epoch` $\times$ `batch_size` turns of data at the beginning. For each warmup iteration, we compute the GAE objective with current critic, sample one tenth of the collected data, train the critic, and repeat this process for `w_iter` iterations. We select `w_epoch = 40` and `w_iter = 5` in our experiments, and make sure that the critic loss converges to a small value before fine-tuning the actor.

B.3.4 KL-Divergence in Reward

Adding a KL-divergence term $KL(\pi \mid \pi_0)$ in reward stabilizes training. Without it, the policy quickly drifts from π_0 and converges to poor solutions. KL penalty encourage local exploration around π_0 before divergence. We observe an interesting observation related to the KL-Divergence:

- **Action Hacking:** The LLM's output can be decomposed into a reasoning path and a final action. We plot the average KL-divergence between π and π_0 for both the reasoning path tokens and the final action tokens. A common failure mode in Crafter arises when the KL divergence of the final action tokens increases significantly faster than that of the reasoning path tokens. In this case, the agent learns to exploit easily accessible rewards early in training by modifying only the final action, without meaningfully improving its underlying reasoning. This leads to poor exploration.

B.4 Conclusion

Verlog solves most of the engineering challenges in building LLM agents for long-horizon, multi-turn tasks. Moving forward, we hope to use this framework to explore core research problems in LLM agents, such as memory design, exploration strategies, value function learning, and handling off-policyness.