THE COST OF FAILURE: QUANTIFYING AND MITIGATING THE IMPACT OF TRAJECTORY ERRORS ON AGENT ROBUSTNESS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Model (LLM) agents solve long-horizon tasks by repeatedly acting, observing feedback, and conditioning later decisions on the accumulated interaction history. This trial-and-error process creates a *Cost of Failure*: invalid actions, execution errors, and low-value feedback persist in the context, polluting later decision making, though models try to recover from mistakes. We quantify this effect with Error Token Density (ETD), controlled context interventions, and failure-conditioned evaluations. On natural ALFWorld rollouts from Qwen3-8B, success drops from 71% with no error-token accumulation to 6.7% when error tokens occupy more than 40% of the trajectory context. Controlled interventions provide causal evidence that accumulated failure context substantially reduces task success, while removing it improves performance. Across ReTool, OpenThinkImg, and ALFWorld, the same failure-conditioned gap appears even for advanced closed-source models. To reduce this failure-induced context pollution, we propose Tool-Feedback Advantage Shaping (TFAS), which combines an accuracy advantage with a tool-feedback advantage estimated from trajectory-level tool-use quality. Unlike outcome-only reinforcement learning, this objective preserves final task credit while separately discouraging invalid intermediate actions. Experiments across three environments and both 4B/8B text and vision-language models show that TFAS achieves the best overall accuracy in all six model-environment settings; notably, on ALFWorld with Qwen3-8B, it improves outcome-only RL from 31.2% to 41.2% overall accuracy and from 21.0% to 52.4% accuracy on error-containing trajectories.

1 INTRODUCTION

Large Language Models (LLMs) Brown et al. (2020) have enabled autonomous agents that interleave reasoning, tool use, and environmental feedback over long-horizon tasks, as in ReAct Yao et al. (2023), Toolformer Schick et al. (2023), and ALFWorld Shridhar et al. (2021). A central challenge for such agents is maintaining reliable decisions as the interaction history grows. Recent agent paradigms therefore emphasize in-context self-correction: the model observes failures or tool feedback, reflects on them, and attempts a revised action in the next turn, as in Reflexion Shinn et al. (2023) and Self-Refine Madaan et al. (2023). This trial-and-error loop is powerful, but it also means that failed actions and their feedback remain inside the same context from which future actions are decoded.

However, this creates a distinct robustness issue: in an autoregressive agent, a failed step persists in the interaction history. Failed actions such as invalid tool calls, unavailable environment actions, exception messages, and low-value feedback remain in the prompt after they occur. Prior work shows that language models can struggle to use long contexts reliably Liu et al. (2024) and can be distracted by irrelevant context Shi et al. (2023); in agent rollouts, accumulated failure traces provide exactly this kind of distracting context. They can compete with useful progress information and make later decisions depend on the agent's own erroneous history. We call this failure-induced degradation of future decisions the *Cost of Failure*.

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.

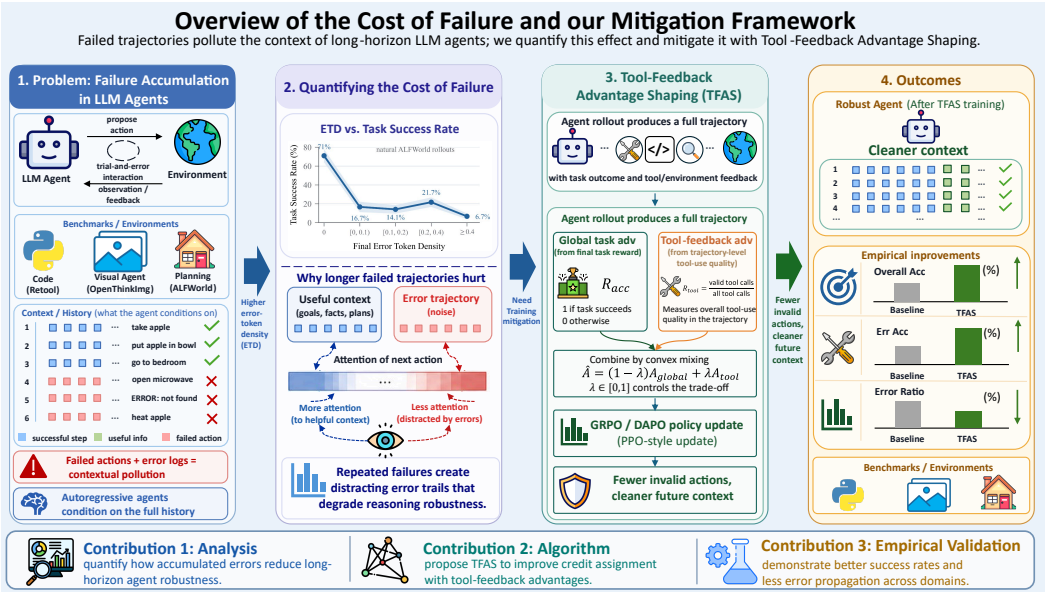

Figure 1: **Overview of the paper.** Failed actions are appended to the autoregressive context and can pollute future decisions. We quantify this Cost of Failure with Error Token Density, show how outcome-only RL misassigns credit to polluting actions, and mitigate the problem with Tool-Feedback Advantage Shaping (TFAS). The resulting policy is validated across planning, code/tool-use, and visual reasoning environments.

For the analysis of *Cost of Failure*, prior work has studied self-correction mechanisms such as Reflexion Shinn et al. (2023) and Self-Refine Madaan et al. (2023), as well as agent failure modes in AgentBoard Ma et al. (2024) and AgentErrorBench/AgentDebug Zhu et al. (2025), but has not directly quantified how much failed interaction content pollutes the autoregressive context. We address this gap with Error Token Density (ETD), a trajectory-level diagnostic that measures the fraction of an interaction history occupied by invalid actions, execution errors, parsing failures, and feedback that mainly records failed transitions. On natural ALFWorld rollouts from Qwen3-8B Yang et al. (2025), task success falls from 71% with no error-token accumulation to 21.7% when ETD is between 0.2 and 0.4, and to 6.7% once error tokens occupy more than 40% of the trajectory context. On matched ALFWorld prefixes with fixed environment states and remaining budgets, failure insertions reduce success by an additional 1.9–3.8 percentage points relative to token-matched neutral insertions, while deleting existing failure turns yields complementary gains. We then broaden the analysis with a binary failure-conditioned split across code/tool-use reasoning, visual reasoning, and embodied planning environments: for example, Qwen3-8B Yang et al. (2025) on ALFWorld drops from 71.0% success on error-free trajectories to 20.2% on trajectories with errors. Even an advanced model such as GPT-5.1 OpenAI (2025b) drops from 80.5% to 33.3%. Together, the controlled and observational results turn the Cost of Failure from an intuitive concern into a concrete training concern: robust agents should avoid injecting invalid or low-value feedback into the context in the first place.

However, outcome-only Reinforcement Learning (RL) provides too coarse a signal to target failure-induced context pollution. Its final reward gives only trajectory-level success or failure, without assigning credit to intermediate actions that preserve or pollute the context. Some recent agent frameworks try to avoid error propagation through external recovery mechanisms such as stepwise rollback Li et al. (2025a); Ma et al. (2025); Li et al. (2025b), but such mechanisms mainly revise or discard failed steps at inference time rather than improving the model's own ability to avoid failed actions. To reduce such error injection during training, we propose *Tool-Feedback Advantage Shaping (TFAS)*, an advantage-level mixing method that combines the accuracy advantage with a separate tool-feedback advantage estimated from trajectory-level tool-use quality. TFAS normalizes task success and tool validity separately and mixes them at the advantage rather than reward level. This preserves the final task objective while giving policy optimization an explicit signal for discouraging context-polluting actions. We validate TFAS across three environments, two modalities, and

two model scales: ReTool and ALFWorld with text-only Qwen3-4B/8B models, and OpenThinkImg with Qwen3-VL-4B/8B vision-language models. TFAS achieves the best overall accuracy in all six model-environment settings; for instance, it improves Qwen3-8B Yang et al. (2025) from 51.3% to 58.4% on ReTool and from 31.4% to 41.2% on ALFWorld. It also reduces error proportions in most settings, such as lowering ReTool Qwen3-8B Yang et al. (2025) from 38.4% under outcome-only RL to 28.5%.

In summary, our main contributions are as follows:

- **Quantifying the Cost of Failure:** We introduce ETD and failure-conditioned evaluation to quantify accumulated errors, and controlled context interventions to provide causal evidence of their impact on agent robustness.
- **Tool-Feedback Advantage Shaping (TFAS):** We propose an RL objective that mixes accuracy advantage with tool-feedback advantage computed from trajectory-level tool-use quality, giving policy optimization an explicit signal to discourage context-polluting actions.
- **Empirical Validation Across Domains:** We evaluate across ReTool, OpenThinkImg, and ALFWorld with text-only and vision-language models. Compared with outcome-only and tool-valid reward baselines, our method improves overall success and reduces error-induced contextual pollution.

2 Related Work

LLM Agents and Error Analysis. LLM agents extend single-turn language models into interactive decision makers that repeatedly act on accumulated environmental feedback Yao et al. (2023); Schick et al. (2023); Shridhar et al. (2021). ReAct Yao et al. (2023) and Toolformer Schick et al. (2023) establish reasoning-action and tool-use paradigms, while ALFWorld Shridhar et al. (2021), WebShop Yao et al. (2022), WebArena Zhou et al. (2024), VisualWebArena Koh et al. (2024), and AgentGym Xi et al. (2024) broaden evaluation across embodied, web, multimodal, and multi-turn settings. Failure-oriented work uses feedback for self-correction Shinn et al. (2023); Madaan et al. (2023), tracks partial progress Ma et al. (2024), and analyzes cascading errors Zhu et al. (2025). These studies focus mainly on detecting or recovering from errors after they occur; less attention has been paid to accumulated failure content as a burden on future context. We treat invalid actions and error feedback as measurable context pollution, complementing evidence that long or irrelevant context degrades language-model behavior Liu et al. (2024); Shi et al. (2023).

Reinforcement Learning for Agents. RL optimizes decisions from environmental feedback rather than static demonstrations Sutton & Barto (2018); Schulman et al. (2017); Ouyang et al. (2022). Building on PPO Schulman et al. (2017), GAE Schulman et al. (2016), and RLHF Ouyang et al. (2022), recent work extends RL to tool use, web interaction, stateful control, and multi-environment training Feng et al. (2025); Qi et al. (2025); Chen et al. (2025); Luo et al. (2025); Xi et al. (2025). Longer rollouts, however, contain invalid actions and execution errors that can interfere with later decisions. Related studies examine multi-turn instability Wang et al. (2025) and inference-time rollback or debugging Li et al. (2025a); Ma et al. (2025); Li et al. (2025b), which revise failures after they enter the trajectory. Conventional reward shaping adds auxiliary signals before advantage estimation Xie et al. (2024), while process supervision commonly learns verifiers from step-level labels Lightman et al. (2023). TFAS instead targets the training signal: it uses rule-checkable execution validity from tools or environments, requires neither a learned reward model nor step labels, and mixes separately normalized task and tool signals only at the advantage level while preserving final task credit. This differs from inference-time recovery, which changes trajectories after failure, and from learned process rewards, which require an additional supervision model; execution validity is already exposed during rollout and adds no annotation pipeline.

3 The Cost of Failure

Modern LLM agents condition each new action on the accumulated interaction history. Failed actions and error feedback therefore remain in the prompt and can distract later reasoning. We call

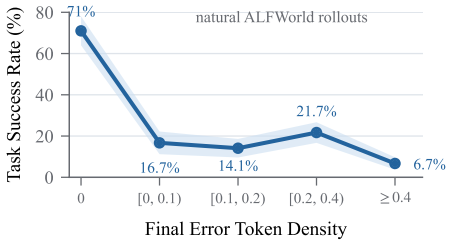

Figure 2: **Task success rate across final-ETD buckets for Qwen3-8B on natural ALFWorld rollouts.** Higher Error Token Density corresponds to a larger fraction of the accumulated context being occupied by invalid actions and error feedback, and is associated with lower task success.

Table 1: **Controlled failure-context interventions on ALFWorld.** Values are changes in success rate (percentage points). Each insertion or deletion is compared with a matched trajectory with an unmodified history.

Intervention	k	$\rho = 0.25$	$\rho = 0.50$
Error insertion (E_k)	1	-4.7	-4.3
	2	-5.7	-6.0
	3	-8.2	-8.8
Neutral insertion (N_k)	1	-0.9	-2.3
	2	-3.8	-3.0
	3	-5.6	-5.9
Failure deletion (D_k)	1	$+2.5$	$+2.0$
	2	$+20.0$	$+13.6$
	3	–	–

this autoregressive degradation the *Cost of Failure*. In this section, we measure failure-induced context with Error Token Density (ETD), test its effect through controlled context interventions, and examine whether the same robustness pattern appears across domains.

3.1 Failure Histories as Autoregressive Context

At step t, the environment provides an observation o_t, and the policy π_θ generates an action a_t from the interaction history. Unlike conventional policies with compact states, an LLM agent conditions on a growing text trajectory:

$$\tau_t = (o_1, a_1, o_2, a_2, \ldots, o_t).$$

Autoregressive action generation therefore depends on the previous context:

$$P(a_t \mid \tau_{t-1}) = \prod_{i=1}^{|a_t|} P(x_{t,i} \mid \tau_{t-1}, x_{t,<i}),$$

where $x_{t,i}$ is the i-th token of the action. We conceptually decompose the history into useful progress tokens τ_{succ} and error tokens τ_{err}. The latter includes invalid tool calls, unavailable environment actions, parsing failures, exception messages, and observations that provide feedback associated with failed transitions. This decomposition is not a new environment state; it is a diagnostic lens for measuring how much of the prompt is occupied by useful progress versus failure-induced noise.

Self-correction methods implicitly assume that τ_{err} provides a useful negative constraint. In practice, long error histories can also act as distractors: they compete with task-relevant evidence, dilute useful progress in the context, and make subsequent decoding depend on the agent's own failed attempts.

3.2 Measuring Failure Context with ETD

To quantify how much failure context appears in real interaction histories, we use Error Token Density (ETD), defined as the fraction of the current prompt occupied by error tokens. Here $|\tau_{err}|$ and $|\tau_{succ}|$ denote the number of tokens in the error and successful-progress portions of the accumulated textual history, respectively:

$$\text{ETD} = \frac{|\tau_{err}|}{|\tau_{succ}| + |\tau_{err}|}.$$

where ETD is a trajectory-level diagnostic for measuring how much of an agent's accumulated context consists of invalid actions, execution errors, parsing failures, and low-value feedback that mainly records failed transitions.

We compute ETD directly on naturally generated ALFWorld rollouts from Qwen3-8B Yang et al. (2025). For each completed trajectory, we tokenize the accumulated interaction history, mark tokens

belonging to invalid actions and their associated error feedback, and bucket trajectories by final ETD. This aims to examine, for a fixed agent and environment, how the amount of accumulated failure context correlates with eventual task completion.

As shown in Figure 2, task success follows an overall downward trend as final ETD increases. Qwen3-8B Yang et al. (2025) succeeds on 71% of trajectories with no error-token accumulation, but success falls to 16.7% when ETD is below 0.1, 14.1% for ETD between 0.1 and 0.2, 21.7% for ETD between 0.2 and 0.4, and only 6.7% once error tokens occupy more than 40% of the trajectory context. This graded association motivates a controlled intervention that separates failure content from task difficulty and general context growth.

3.3 Controlled Failure-Context Interventions

To further examine causality, we conduct controlled failure-context insertion and deletion experiments. At normalized trajectory anchors $\rho \in \{0.25, 0.5\}$, we branch from the same unfinished prefix and environment state while keeping the remaining action budget fixed. For insertion, E_k adds $k \in \{1, 2, 3\}$ state-preserving invalid-action/error-feedback turns, whereas N_k adds the same number of neutral turns with the same token count as the corresponding error content. For deletion, D_k removes the most recent k complete failure turns from a prefix that already contains errors, and we compare it with a matched trajectory with an unmodified history.

Table 1 reveals a substantial, cumulative cost of failure context. From the Qwen3-8B baseline, inserting one failure turn reduces success by 4.3–4.7 points, while three turns reduce it by 8.2–8.8 points—approximately one-third of the baseline performance. Error insertions are consistently 1.9–3.8 points worse than token-matched neutral insertions, showing that context length alone does not explain the degradation. The reverse intervention gives complementary evidence: deleting one failure turn recovers 2.0–2.5 points, and deleting two yields larger gains of 13.6–20.0 points. These bidirectional results demonstrate the pronounced downstream impact of retaining failure content in the agent context.

3.4 Failure-Conditioned Robustness Across Domains

The controlled interventions isolate the effect of failure content inside ALFWorld. We next examine whether the corresponding observational robustness gap appears more broadly, using a binary split between trajectories that contain at least one error and trajectories that remain error-free. We evaluate this split on three representative agent domains: ReTool Feng et al. (2025) for code-interpreter-augmented mathematical reasoning, OpenThinkImg Su et al. (2025) for visual tool reasoning, and ALFWorld Shridhar et al. (2021) for embodied planning; detailed environment descriptions and training settings are provided in Section 5. Table 2 reports this split for the three domains and shows whether the same robustness gap appears across different agent settings. The evaluated models are referenced to their corresponding documentation: GPT-5 OpenAI (2025a), GPT-5.1 OpenAI (2025b), Claude-Sonnet-4 Anthropic (2025), Gemini-3-Flash Google DeepMind (2025), Doubao-Seed-1.6 ByteDance Seed Team (2025), DeepSeek-V3.2 DeepSeek-AI (2025), Qwen3 Yang et al. (2025), and Qwen3-VL Bai et al. (2025).

Across the three domains, error-containing trajectories usually have substantially lower success rates than clean trajectories, even for strong models. In ReTool, GPT-5 drops from 90.9% success on error-free trajectories to 61.5% on trajectories with errors, while Qwen3-8B Yang et al. (2025) drops from 68.3% to 49.3%. The gap is especially large in ALFWorld, where each wrong move is appended to a long interaction history: Qwen3-8B Yang et al. (2025) falls from 71.0% to 20.2%, and GPT-5.1 falls from 80.5% to 33.3%. OpenThinkImg shows that error frequency alone can understate the problem: for Qwen3-VL-8B-Instruct Bai et al. (2025), only 1.4% of trajectories contain errors, but those trajectories succeed only 16.2% of the time, far below the 73.8% success rate on error-free trajectories. These results support the same conclusion as ETD: once errors enter the context, task completion becomes substantially harder, so training should reduce the chance of injecting such errors in the first place. The qualitative example in Figure 3 shows the same pattern at the trajectory level: after an unavailable desklamp action is appended to the history, the baseline agent repeatedly conditions on the resulting error feedback and fails to return to the valid path.

Table 2: **Failure-conditioned performance across three agent domains.** SR: success rate; Prop.: trajectory proportion; Gap: error-free minus error-containing SR. Errors consistently predict lower success.

Environment	Model	Overall	With Error		Without Error		Gap
		SR (%)	SR (%)	Prop. (%)	SR (%)	Prop. (%)	Δ SR (%)
ReTool	GPT-5 OpenAI (2025a)	65.5	61.5	19.1	90.9	80.9	29.4
	Claude-Sonnet-4 Anthropic (2025)	52.1	40.2	46.7	62.5	53.3	22.3
	Gemini-3-Flash Google DeepMind (2025)	46.7	28.5	23.3	56.2	76.7	27.7
	DeepSeek-V3.2 DeepSeek-AI (2025)	76.5	50.0	1.2	91.6	98.8	**41.6**
	Qwen3-8B Yang et al. (2025)	51.3	49.3	30.0	68.3	70.0	19.0
OpenThinkImg	GPT-5 OpenAI (2025a)	48.9	19.8	16.0	71.4	84.0	51.6
	Claude-Sonnet-4 Anthropic (2025)	79.9	39.5	25.2	60.5	74.8	21.0
	Gemini-3-Flash Google DeepMind (2025)	82.7	38.9	28.1	61.1	71.9	22.2
	Doubao-Seed-1.6 ByteDance Seed Team (2025)	73.1	10.0	2.3	75.7	97.7	**65.7**
	Qwen3-VL-8B-Instruct Bai et al. (2025)	48.8	16.2	1.4	73.8	98.6	57.6
ALFWorld	GPT-5.1 OpenAI (2025b)	76.4	33.3	9.4	80.5	90.6	47.2
	Claude-Sonnet-4 Anthropic (2025)	39.3	27.8	77.1	78.1	22.9	50.3
	Gemini-3-Flash Google DeepMind (2025)	59.3	42.9	5.0	60.2	95.0	17.3
	DeepSeek-V3.2 DeepSeek-AI (2025)	36.4	20.6	24.3	41.5	75.7	20.9
	Qwen3-8B Yang et al. (2025)	31.4	20.2	17.0	71.0	83.0	**50.8**

Note. Prop. is computed over trajectories with at least one tool invocation; trajectories where the model does not call a tool are included in Overall SR but excluded from the error-conditioned proportion split.

4 METHOD

Section 3 shows that failed actions can pollute the future autoregressive context. We therefore improve PPO-style policy optimization algorithms, such as GRPO Shao et al. (2024) and DAPO Yu et al. (2025), so that policy updates reflect both final task success and the quality of tool use within the trajectory.

4.1 TOOL-FEEDBACK ADVANTAGE SHAPING (TFAS)

Outcome-only GRPO/DAPO derives an accuracy advantage A^{acc} from binary task success and assigns it across the trajectory. It therefore cannot distinguish an invalid call inside a successful trajectory from the valid actions that complete the task. Tool-Feedback Advantage Shaping (TFAS) retains this task-level credit while adding a separately normalized signal for tool-use quality. Let $c_t \in \{0, 1\}$ indicate whether step t invokes a tool, and let $v_t \in \{0, 1\}$ indicate whether that invocation is valid and executable. We aggregate all tool-use outcomes in the completed trajectory as:

$$R^{tool}(\tau) = \begin{cases} \frac{\sum_{t=1}^{T} c_t v_t}{\sum_{t=1}^{T} c_t}, & \sum_{t=1}^{T} c_t > 0, \\ 0, & \sum_{t=1}^{T} c_t = 0. \end{cases}$$

Thus $R^{tool}(\tau) \in [0, 1]$ is a single trajectory-level score: the normalized fraction of correct tool calls within the completed trajectory. We convert this trajectory-level tool reward into a tool-feedback advantage using the same group-relative normalization as the task reward. For a prompt group $\mathcal{G}$, let $\mu_{\mathcal{G}}^{tool}$ and $\sigma_{\mathcal{G}}^{tool}$ be the mean and standard deviation of R^{tool} over the sampled trajectories in that group. The tool-feedback advantage for trajectory τ is

$$A^{tool}(\tau) = \frac{R^{tool}(\tau) - \mu_{\mathcal{G}}^{tool}}{\sigma_{\mathcal{G}}^{tool} + \epsilon}.$$

We then broadcast this scalar advantage to the response tokens of the corresponding trajectory, following the same token-level masking used by GRPO/DAPO.

4.2 MIXED ADVANTAGE OBJECTIVE

The final optimization advantage mixes task-accuracy and tool-use credit:

$$\hat{A}_t = (1 - \lambda) A_t^{acc} + \lambda A_t^{tool}, \tag{1}$$

where $\lambda \in [0, 1]$ controls the strength of tool-feedback shaping. When $\lambda = 0$, the objective reduces to the corresponding outcome-only policy optimization algorithm.

Table 3: **Main Experimental Results across Three Environments.** We report the Overall Accuracy (Overall Acc), the proportion of trajectories that contain at least one error (Err Proportion), and the task success rate for trajectories that contain at least one error step (Err Acc). The untrained base model is included as the baseline. The best results are highlighted in **bold**. TFAS achieves the best overall accuracy in all settings, demonstrating the effectiveness of our method.

Environment	Model	Reward Strategy	Overall Acc (%)↑	Err Proportion (%)↓	Err Acc (%)↑
ReTool	Qwen3-4B Yang et al. (2025)	Base Model (No Training)	47.9	40.3	23.7
		Original Outcome Reward	50.4	41.4	22.7
		+ Tool Valid Reward	52.5	38.8	19.1
		+ **TFAS (Ours)**	**54.7**	**34.6**	**33.3**
	Qwen3-8B Yang et al. (2025)	Base Model (No Training)	51.3	30.0	49.3
		Original Outcome Reward	55.1	38.4	26.2
		+ Tool Valid Reward	56.2	31.2	25.4
		+ **TFAS (Ours)**	**58.4**	**28.5**	**53.6**
OpenThinkImg	Qwen3-VL-4B Bai et al. (2025)	Base Model (No Training)	42.4	7.8	15.5
		Original Outcome Reward	44.3	7.1	**19.4**
		+ Tool Valid Reward	46.5	5.7	9.5
		+ **TFAS (Ours)**	**49.3**	**3.8**	10.7
	Qwen3-VL-8B Bai et al. (2025)	Base Model (No Training)	48.8	**1.4**	16.2
		Original Outcome Reward	49.2	12.6	18.4
		+ Tool Valid Reward	49.5	2.7	16.7
		+ **TFAS (Ours)**	**52.9**	2.2	**20.9**
ALFWorld	Qwen3-4B Yang et al. (2025)	Base Model (No Training)	17.9	83.6	10.3
		Original Outcome Reward	17.8	84.2	12.3
		+ Tool Valid Reward	20.9	66.7	22.1
		+ **TFAS (Ours)**	**24.3**	**64.1**	**31.8**
	Qwen3-8B Yang et al. (2025)	Base Model (No Training)	31.4	17.0	20.2
		Original Outcome Reward	31.2	18.3	21.0
		+ Tool Valid Reward	33.6	18.0	35.7
		+ **TFAS (Ours)**	**41.2**	**15.8**	**52.4**

We plug $\hat{A}_t$ into the policy-ratio surrogate used by PPO-style algorithms:

$$\mathcal{L}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right],$$

where $r_t(\theta)$ is the standard PPO policy ratio between the current and behavior policies. Unlike reward-level summation, advantage-level mixing separately normalizes task success and tool quality, providing a direct gradient signal.

5 Experiment

In this section, we describe the experimental setup, compare TFAS with outcome-only and tool-valid reward baselines, and analyze its effects on task success, error-containing trajectories, and the advantage mixing weight.

5.1 Experimental Setup

Environments and Datasets. To demonstrate the generalizability of our algorithm across different state spaces and error modalities, we evaluate our approach on three distinct long-horizon interactive environments: (1) **ReTool** Feng et al. (2025): a code-interpreter-augmented mathematical reasoning setting where agents solve competition-style math problems by interleaving natural-language reasoning with executable Python code; (2) **OpenThinkImg** Su et al. (2025): a tool-augmented visual reasoning framework where VLM agents solve chart reasoning tasks by adaptively invoking external vision tools such as OCR and image zooming; and (3) **ALFWorld** Shridhar et al. (2021): a text-based embodied simulation where multi-step household tasks naturally generate exploration-driven trial-and-error noise. Together, these environments provide a comprehensive testbed for evaluating our method's ability to mitigate contextual interference across code-tool reasoning, visual-linguistic, and physical domains.

Baselines. We compare our method across two categories of baseline models to comprehensively evaluate its effectiveness and generalizability: (1) pure text models, specifically Qwen3-4B Yang et al. (2025) and Qwen3-8B Yang et al. (2025); and (2) Vision-Language Models (VLMs), specifically Qwen3-VL-4B Bai et al. (2025) and Qwen3-VL-8B Bai et al. (2025). This setup tests whether

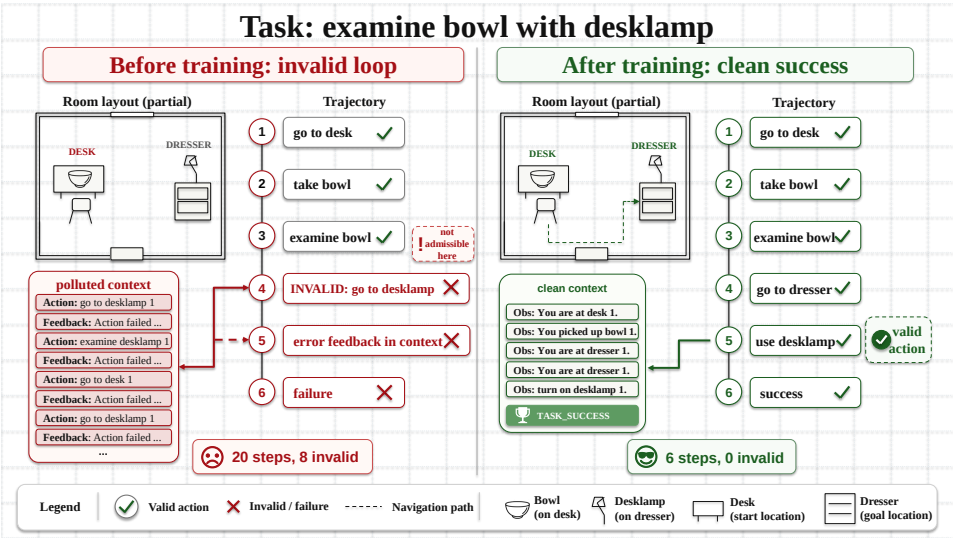

Figure 3: **Qualitative ALFWorld case comparing behavior before and after TFAS training.** The baseline policy enters an invalid-action loop, while the trained policy follows executable actions and completes the task without invalid steps.

the proposed method generalizes across both modalities (text-only and vision-language) and model scales (4B and 8B).

Reward Strategies. To systematically evaluate the effectiveness of our proposed mechanism, we compare three distinct reinforcement learning reward strategies within our training framework: (1) **Original Outcome Reward (Baseline)**: This standard setup relies solely on the final accuracy reward (acc reward) upon task success. (2) **Tool Validation Reward**: This strategy augments the final accuracy reward with a trajectory-level tool-valid reward, computed as the proportion of correct tool-use steps over the completed rollout. (3) **Tool-Feedback Advantage Shaping (TFAS, Ours)**: TFAS converts the same tool-valid signal into a separate advantage and mixes it with the accuracy advantage through Equation 1. Both tool-feedback strategies use the same tool-signal weight.

5.2 Experiment Results

Table 3 provides several consistent conclusions across the three environments. We summarize the key findings below.

Higher Overall Acc. Our TFAS achieves the highest Overall Acc in all six model-environment settings, over the base model, outcome-only RL, and tool-valid reward baselines. For example, it improves Qwen3-8B Yang et al. (2025) from 51.3% to 58.4% on ReTool and from 31.4% to 41.2% on ALFWorld. This confirms that TFAS improves final task completion rather than only changing error statistics.

Lower Err Proportion. Our method usually reduces the proportion of error-containing trajectories. It obtains the lowest Err Proportion in ReTool and ALFWorld for both model sizes. In OpenThinkImg, for Qwen3-VL-8B Bai et al. (2025), it also sharply reduces the error proportion compared with outcome-only RL, from 12.6% to 2.2%. This suggests that TFAS generally suppresses the formation of polluted contexts before they accumulate.

Stronger Err Acc. Error-conditioned accuracy measures the task-success rate after at least one error has already entered the trajectory. TFAS gives the best Err Acc in ReTool and ALFWorld, reaching 53.6% on ReTool with Qwen3-8B Yang et al. (2025) and 52.4% on ALFWorld with Qwen3-8B Yang et al. (2025). Together with the Err Proportion reductions observed in many settings, this suggests that the method reduces harmful error injection while also making the remaining error-containing trajectories less likely to collapse.

Tool Feedback Matters. Our results suggest that tool-failure feedback is more effective when used beyond a plain reward bonus. Outcome-only RL often provides limited gains, while Tool Valid

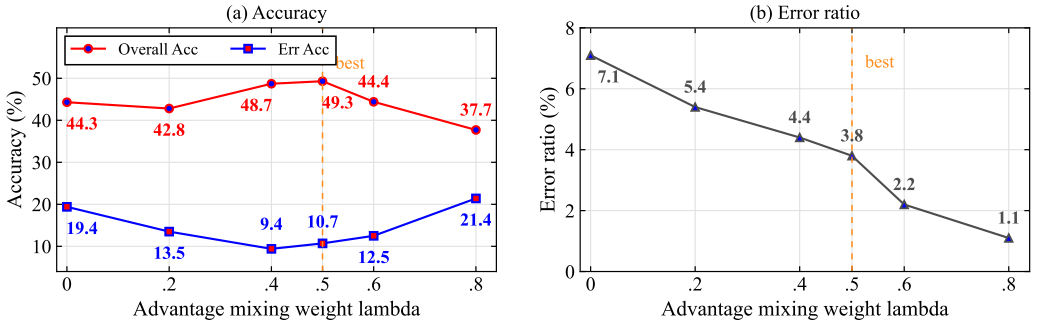

Figure 4: **Sensitivity analysis of the advantage mixing weight λ on OpenThinkImg with Qwen3-VL-4B.** (a) Overall Acc under different mixing weights. (b) Err Ratio under the same settings. A moderate λ provides the best balance between task success and error reduction.

Reward sometimes lowers Err Acc, such as in ReTool and OpenThinkImg with Qwen3-VL-4B Bai et al. (2025). Mixing tool feedback at the advantage level gives a clearer optimization signal than either ignoring failures or adding a shallow validity reward. Thus, the benefit comes from how tool feedback is injected into optimization, not merely from adding an auxiliary reward.

Broad Applicability. The same pattern holds across different tasks, modalities, and model sizes: text-only agents on ReTool and ALFWorld, VLM agents on OpenThinkImg, and both 4B and 8B scales. This suggests that TFAS addresses a general failure mode of agent training, and failure-induced context pollution is a cross-domain agent-training problem.

5.3 Ablation: Advantage Mixing Weight

Figure 4 studies the advantage mixing weight λ on OpenThinkImg with Qwen3-VL-4B Bai et al. (2025), keeping the final accuracy reward, policy-optimization configuration, and training budget fixed. The $\lambda = 0$ setting corresponds to outcome-only optimization, while larger values increase the contribution of tool-feedback advantage in Equation 1. Adding tool-use feedback improves overall task success and reduces error-containing trajectories, with $\lambda = 0.5$ giving the best balance in this setting. When λ becomes too large, the tool signal can dominate the global task objective and reduce final accuracy, indicating that tool-feedback advantage is most effective as a complementary shaping signal rather than a replacement for task-level credit. The best λ may vary across environments and models; the values used in this paper are reported in Appendix A.

5.4 Qualitative Analysis

Figure 3 shows the ALFWorld case behind the gains in Table 3. The baseline reaches the desk, takes the bowl, then attempts an unavailable action; the resulting error feedback triggers repeated invalid actions. After TFAS training, the agent follows the executable layout and completes the task in 6 steps without invalid actions. This illustrates how TFAS assigns lower credit to actions that inject invalid feedback and higher credit to actions that preserve useful state evidence; similar examples appear in Appendix F.

6 Conclusion

This paper studies the Cost of Failure in long-horizon LLM agents; controlled insertions and deletions show that failed actions and feedback degrade later decisions beyond context growth. Tool-Feedback Advantage Shaping (TFAS) mixes accuracy and rule-checkable tool-feedback advantages while preserving final task credit. Across planning, tool-use, and visual reasoning, TFAS improves overall success and often reduces error-containing trajectories, showing that robust training should reward completion while discouraging harmful feedback.

AI USE STATEMENT

In this work, we used AI only to edit the paper for grammar, wording, and readability. We did not use generative AI to generate data, develop the method or conceptual framework, formulate mathematical claims, design or run experiments, implement software, analyze or interpret results, create figures, or generate scientific claims. All AI-assisted edits were reviewed and verified by the authors, who take full responsibility for the final manuscript and all reported results.

ETHICS STATEMENT

This work uses established benchmark environments and publicly documented models and datasets; it introduces no human-subject data or personally identifiable information. The method is intended to reduce invalid tool actions and improve agent reliability, but deployment should still use appropriate tool permissions, monitoring, and human oversight for consequential tasks.

REPRODUCIBILITY STATEMENT

We provide the TFAS core algorithm, dataset and evaluation splits, training hyperparameters, model configurations, compute details, and independent repeated runs with confidence intervals in the appendix. The reported experiments use fixed environment settings and explicitly documented rollout and optimization configurations to support reproduction.

REFERENCES

Anthropic. System card: Claude Opus 4 and Claude Sonnet 4, 2025. URL https://assets.anthropic.com/m/6c940a1b69ed6a1c/original/Claude-4-System-Card.pdf.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025. URL https://arxiv.org/abs/2511.21631.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

ByteDance Seed Team. Introduction to techniques used in Seed1.6, 2025. URL https://seed.bytedance.com/en/blog/introduction-to-techniques-used-in-seed1-6.

Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. Reinforcement learning for long-horizon interactive LLM agents. *arXiv preprint arXiv:2502.01600*, 2025. URL https://arxiv.org/abs/2502.01600.

DeepSeek-AI. DeepSeek-V3.2: Pushing the frontier of open large language models, 2025. URL https://arxiv.org/abs/2512.02556.

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, 2025. URL https://arxiv.org/abs/2504.11536.

Google DeepMind. Gemini 3 Flash, 2025. URL https://deepmind.google/technologies/gemini/flash/.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://aclanthology.org/2024.acl-long.50/.

Xingzuo Li, Kehai Chen, Yunfei Long, Xuefeng Bai, Yong Xu, and Min Zhang. Generator-assistant stepwise rollback framework for large language model agent. *arXiv preprint arXiv:2503.02519*, 2025a. URL https://arxiv.org/abs/2503.02519.

Yang Li, Siqi Ping, Xiyu Chen, Xiaojian Qi, Zigan Wang, Ye Luo, and Xiaowei Zhang. AgentGit: A version control framework for reliable and scalable LLM-powered multi-agent systems. *arXiv preprint arXiv:2511.00628*, 2025b. URL https://arxiv.org/abs/2511.00628.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023. URL https://arxiv.org/abs/2305.20050.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://arxiv.org/abs/2307.03172.

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. URL https://arxiv.org/abs/2508.03680.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. AgentBoard: An analytical evaluation board of multi-turn LLM agents. In *Advances in Neural Information Processing Systems*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/877b40688e330a0e2a3fc24084208dfa-Abstract-Datasets_and_Benchmarks_Track.html.

Ming Ma, Jue Zhang, Fangkai Yang, Yu Kang, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. DoVer: Intervention-driven auto debugging for LLM multi-agent systems. *arXiv preprint arXiv:2512.06749*, 2025. URL https://arxiv.org/abs/2512.06749.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2303.17651.

Mathematical Association of America. MAA invitational competitions, 2025. URL https://maa.org/maa-invitational-competitions/.

OpenAI. GPT-5 system card, 2025a. URL https://openai.com/index/gpt-5-system-card/.

OpenAI. Introducing GPT-5.1 for developers, 2025b. URL https://openai.com/index/gpt-5-1-for-developers/.

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 F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022. URL https://openreview.net/forum?id=TG8KACxEON.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, et al. WebRL: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2411.02337.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2302.04761.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1506.02438.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

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, 2024. URL https://arxiv.org/abs/2402.03300.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pp. 31210–31227. PMLR, 2023. URL https://proceedings.mlr.press/v202/shi23a.html.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2303.11366.

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. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Zhaochen Su, Linjie Li, Mingyang Song, Yunzhuo Hao, Zhengyuan Yang, Jun Zhang, Guanjie Chen, Jiawei Gu, Juntao Li, Xiaoye Qu, and Yu Cheng. OpenThinkIMG: Learning to think with images via visual tool reinforcement learning, 2025. URL https://arxiv.org/abs/2505.08617.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

verl Project. verl: Volcano engine reinforcement learning for llms, 2024. URL https://github.com/verl-project/verl.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025. URL https://arxiv.org/abs/2504.20073.

Zirui Wang, Mengzhou Xia, Luxi He, Howard Chen, Yitao Liu, Richard Zhu, Kaiqu Liang, Xindi Wu, Haotian Liu, Sadhika Malladi, Alexis Chevalier, Sanjeev Arora, and Danqi Chen. CharXiv: Charting gaps in realistic chart understanding in multimodal LLMs, 2024. URL https://arxiv.org/abs/2406.18521.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, 2020. URL https://aclanthology.org/2020.emnlp-demos.6/.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, et al. AgentGym: Evolving large language model-based agents across diverse environments. *arXiv preprint arXiv:2406.04151*, 2024. URL https://arxiv.org/abs/2406.04151.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Honglin Guo, Jiaqi Liu, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, et al. AgentGym-RL: Training LLM agents for long-horizon decision making through multi-turn reinforcement learning. *arXiv preprint arXiv:2509.08755*, 2025. URL https://arxiv.org/abs/2509.08755.

Tianbao Xie, Soroush Vosoughi, Jun-Yan Zhu, Antonio Torralba, William T. Freeman, and Yuandong Tian. Text2Reward: Reward shaping with language models for reinforcement learning. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=tUM39YTRxH.

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. URL https://arxiv.org/abs/2505.09388.

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, pp. 20744–20757, 2022. URL https://openreview.net/forum?id=R9KnuFlvnU.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. PyTorch FSDP: Experiences on scaling fully sharded data parallel. *Proceedings of the VLDB Endowment*, 16(12): 3848–3860, 2023. URL https://arxiv.org/abs/2304.11277.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2307.13854.

Kunlun Zhu, Zijia Liu, Bingxuan Li, Muxin Tian, Yingxuan Yang, Jiaxun Zhang, Pengrui Han, Qipeng Xie, Fuyang Cui, Weijia Zhang, et al. Where LLM agents fail and how they can learn from failures. *arXiv preprint arXiv:2509.25370*, 2025. URL https://arxiv.org/abs/2509.25370.

A Training Hyperparameters

In this section, Tables 4–9 report the hyperparameter settings used for all reinforcement-learning experiments in Section 5. We list the configurations separately for each environment and model size, covering ReTool, OpenThinkImg, and ALFWorld. Within each table, the three columns correspond to the outcome-only reward baseline, the tool-valid reward baseline, and our Tool-Feedback Advantage Shaping (TFAS) strategy, respectively. Unless otherwise specified, hyperparameters are kept identical across reward strategies within the same environment-model setting to ensure a controlled comparison. We use GRPO Shao et al. (2024) and DAPO Yu et al. (2025) as the underlying PPO-style optimizers.

B Method Details

B.1 Core Codes for TFAS

In this section, we show the core codes of TFAS:

Listing 1: Core code for Tool-Feedback Advantage Shaping.

```
"""
Dual Advantage for Multi-Turn GRPO

Key idea:
   final_adv = (1 - w) * success_advantage + w * valid_fraction_advantage

Instead of mixing two reward signals into one scalar before GRPO
    normalisation,
we normalise each signal independently and blend the resulting advantages
    .
This preserves gradient signal when the success reward has zero variance
(all-fail or all-succeed prompt groups).
"""

from __future__ import annotations
from collections import defaultdict

import numpy as np
import torch


def _grpo_normalise(scores: torch.Tensor, group_ids: list[str],
                eps: float = 1e-6) -> torch.Tensor:
   """Within-group zero-mean, unit-variance normalisation (standard GRPO)
       .
   Groups with zero variance get advantage = 0."""
   adv = torch.empty_like(scores)
   groups: dict[str, list[int]] = defaultdict(list)
   for i, gid in enumerate(group_ids):
      groups[gid].append(i)
   for idxs in groups.values():
      g = scores[idxs]
      mu, std = g.mean(), g.std() if len(idxs) > 1 else torch.tensor(1.0)
      adv[idxs] = (g - mu) / (std + eps) if std > eps else 0.0
   return adv


def compute_dual_advantage(
   rm_scores: torch.Tensor, # [N, seq_len] success reward (0/1 at last
       token)
   response_mask: torch.Tensor, # [N, seq_len] response token mask
   prompt_uids: np.ndarray, # [N] prompt group id per row
   traj_uids: np.ndarray, # [N] trajectory id per row
   is_valid_action: np.ndarray, # [N] 1 if step action was valid
```

```
    valid_fraction_coef: float, # coefficient to scale valid-fraction
        score
    vf_adv_weight: float, # blend weight w (e.g. 0.5)
    eps: float = 1e-6,
) -> tuple[torch.Tensor, dict]:
    """Compute dual advantage with separate GRPO normalisation per signal
        ."""

    # -- Step 1: aggregate per-step rows into per-trajectory scalars
        ----------
    row_scores = rm_scores.sum(dim=-1) # [N]

    traj_to_rows: dict[str, list[int]] = defaultdict(list)
    for i, uid in enumerate(traj_uids):
       traj_to_rows[str(uid)].append(i)

    traj_ids = list(traj_to_rows.keys())
    success_scores, vf_scores, traj_prompts = [], [], []

    for tid in traj_ids:
       rows = traj_to_rows[tid]
       # Success: same binary reward across all steps in a trajectory
       success_scores.append(row_scores[rows[0]])
       # Valid-fraction: proportion of valid actions, scaled by
           coefficient
       vf = float(is_valid_action[rows].sum()) / max(len(rows), 1)
       vf_scores.append(torch.tensor(vf * valid_fraction_coef,
                                     device=rm_scores.device))
       traj_prompts.append(str(prompt_uids[rows[0]]))

    success_t = torch.stack(success_scores)
    vf_t = torch.stack(vf_scores)

    # -- Step 2: GRPO normalise each signal independently
        ----------------------
    # This is the core of dual advantage: each reward signal gets its own
    # within-group normalisation, so neither signal can drown out the
        other.
    success_adv = _grpo_normalise(success_t, traj_prompts, eps)
    vf_adv = _grpo_normalise(vf_t, traj_prompts, eps)

    # -- Step 3: blend
        ------------------------------------------------------------
    blended = (1.0 - vf_adv_weight) * success_adv + vf_adv_weight * vf_adv

    # -- Step 4: broadcast back to per-step rows
        -----------------------------
    advantages = torch.zeros_like(response_mask, dtype=torch.float32)
    for idx, tid in enumerate(traj_ids):
       rows = traj_to_rows[tid]
       advantages[rows] = blended[idx] * response_mask[rows]

    metrics = {
       "dual_adv/success_adv_mean": float(success_adv.mean()),
       "dual_adv/vf_adv_mean": float(vf_adv.mean()),
       "dual_adv/blended_adv_mean": float(blended.mean()),
    }
    return advantages, metrics
```

Table 4: **Hyperparameter settings for Qwen3-4B ReTool experiments.**

	Outcome Reward	Tool Valid Reward	TFAS (Ours)
Pretrained Model	Qwen3-4B-Instruct-2507 Yang et al. (2025)	Qwen3-4B-Instruct-2507 Yang et al. (2025)	Qwen3-4B-Instruct-2507 Yang et al. (2025)
RL Algorithm	DAPO	DAPO	DAPO
Train Batchsize	64	64	64
Temperature	1.0	1.0	1.0
Num Response per Prompt	16	16	16
Max Prompt Length	2048	2048	2048
Max Response Length	16384	16384	16384
Bf16	True	True	True
Actor Learning Rate	1e-6	1e-6	1e-6
KL Loss Coef	0.0	0.0	0.0
Steps	40	40	40
λ	–	0.5	0.5

Table 5: **Hyperparameter settings for Qwen3-8B ReTool experiments.**

	Outcome Reward	Tool Valid Reward	TFAS (Ours)
Pretrained Model	Qwen3-8B Yang et al. (2025)	Qwen3-8B Yang et al. (2025)	Qwen3-8B Yang et al. (2025)
RL Algorithm	DAPO	DAPO	DAPO
Train Batchsize	64	64	64
Temperature	1.0	1.0	1.0
Num Response per Prompt	16	16	16
Max Prompt Length	2048	2048	2048
Max Response Length	16384	16384	16384
Bf16	True	True	True
Actor Learning Rate	1e-6	1e-6	1e-6
KL Loss Coef	0.0	0.0	0.0
Steps	40	40	40
λ	–	0.5	0.5

Table 6: **Hyperparameter settings for Qwen3-VL-4B OpenThinkImg experiments.**

	Outcome Reward	Tool Valid Reward	TFAS (Ours)
Pretrained Model	Qwen3-VL-4B-Instruct Bai et al. (2025)	Qwen3-VL-4B-Instruct Bai et al. (2025)	Qwen3-VL-4B-Instruct Bai et al. (2025)
RL Algorithm	GRPO	GRPO	GRPO
Train Batchsize	64	64	64
Temperature	1.0	1.0	1.0
Num Response per Prompt	8	8	8
Max Prompt Length	4096	4096	4096
Max Response Length	512	512	512
Bf16	True	True	True
Actor Learning Rate	1e-6	1e-6	1e-6
KL Loss Coef	0.0	0.0	0.0
Steps	8	8	8
λ	–	0.5	0.5

Table 7: **Hyperparameter settings for Qwen3-VL-8B OpenThinkImg experiments.**

	Outcome Reward	Tool Valid Reward	TFAS (Ours)
Pretrained Model	Qwen3-VL-8B-Instruct Bai et al. (2025)	Qwen3-VL-8B-Instruct Bai et al. (2025)	Qwen3-VL-8B-Instruct Bai et al. (2025)
RL Algorithm	GRPO	GRPO	GRPO
Train Batchsize	16	16	16
Temperature	1.0	1.0	1.0
Num Response per Prompt	8	8	8
Max Prompt Length	4096	4096	4096
Max Response Length	512	512	512
Bf16	True	True	True
Actor Learning Rate	1e-6	1e-6	1e-6
KL Loss Coef	0.0	0.0	0.0
Steps	8	8	8
λ	–	0.5	0.5

Table 8: **Hyperparameter settings for Qwen3-4B ALFWorld experiments.**

	Outcome Reward	Tool Valid Reward	TFAS (Ours)
Pretrained Model	Qwen3-4B-Instruct-2507 Yang et al. (2025)	Qwen3-4B-Instruct-2507 Yang et al. (2025)	Qwen3-4B-Instruct-2507 Yang et al. (2025)
RL Algorithm	GRPO	GRPO	GRPO
Train Batchsize	64	64	64
Temperature	1.0	1.0	1.0
Num Response per Prompt	8	8	8
Max Prompt Length	12000	12000	12000
Max Response Length	256	256	256
Bf16	True	True	True
Actor Learning Rate	1e-6	1e-6	1e-6
KL Loss Coef	0.15	0.15	0.15
Steps	23	23	23
λ	–	0.17	0.17

Table 9: **Hyperparameter settings for Qwen3-8B ALFWorld experiments.**

	Outcome Reward	Tool Valid Reward	TFAS (Ours)
Pretrained Model	Qwen3-8B Yang et al. (2025)	Qwen3-8B Yang et al. (2025)	Qwen3-8B Yang et al. (2025)
RL Algorithm	GRPO	GRPO	GRPO
Train Batchsize	64	64	64
Temperature	1.0	1.0	1.0
Num Response per Prompt	16	16	16
Max Prompt Length	12000	12000	12000
Max Response Length	256	256	256
Bf16	True	True	True
Actor Learning Rate	1e-6	1e-6	1e-6
KL Loss Coef	0.15	0.15	0.15
Steps	23	23	23
λ	–	0.17	0.17

Table 10: **Training and evaluation data used for each environment.**

Environment	Training Set	Test / Evaluation Set
ReTool	DAPO-Math-17k Yu et al. (2025)	AIME 2025 Mathematical Association of America (2025)
OpenThinkImg	OpenThinkIMG-Chart-RL-14501 Su et al. (2025)	CharXiv Wang et al. (2024)
ALFWorld	ALFWorld Shridhar et al. (2021)	ALFWorld Shridhar et al. (2021)

Table 11: **Independent repeats and 95% CIs for the ALFWorld main experiment.** "Runs" lists the three independent results, and the adjacent column reports the sample mean $\pm$ the two-sided 95% CI half-width. All values are percentages.

Model	Reward Strategy	Overall Acc (↑)		Err Proportion (↓)		Err Acc (↑)	
		Runs 1/2/3	Mean $\pm$ 95% CI	Runs 1/2/3	Mean $\pm$ 95% CI	Runs 1/2/3	Mean $\pm$ 95% CI
Qwen3-4B	Original Outcome Reward	17.8/16.7/17.9	17.5 ± 1.65	84.2/85.2/84.0	84.5 ± 1.60	12.3/12.0/12.0	12.1 ± 0.43
	+ Tool Valid Reward	20.9/21.3/21.0	21.1 ± 0.52	66.7/65.5/66.7	66.3 ± 1.72	22.1/21.8/21.9	21.9 ± 0.38
	+ TFAS (Ours)	24.3/24.5/23.9	24.2 ± 0.76	64.1/63.5/64.5	64.0 ± 1.25	31.8/31.9/31.5	31.7 ± 0.52
Qwen3-8B	Original Outcome Reward	31.2/30.0/32.4	31.2 ± 2.98	18.3/18.9/17.3	18.2 ± 2.01	21.0/20.1/22.5	21.2 ± 3.01
	+ Tool Valid Reward	33.6/33.4/33.0	33.3 ± 0.76	18.0/19.8/18.7	18.8 ± 2.25	35.7/35.4/35.3	35.5 ± 0.52
	+ TFAS (Ours)	41.2/42.1/40.8	41.4 ± 1.65	15.8/14.8/16.5	15.7 ± 2.12	52.4/52.3/52.3	52.3 ± 0.14

B.2 Other Details

We implement our method based on the **verl** reinforcement learning framework, into which we have integrated the aforementioned ALFWorld, ReTool, and OpenThinkImg environments. To facilitate efficient distributed training, we utilize the **FSDP** (Fully Sharded Data Parallel) backend for all experiments. The base models include Qwen3-4B Yang et al. (2025), Qwen3-8B Yang et al. (2025), Qwen3-VL-4B Bai et al. (2025), and Qwen3-VL-8B Bai et al. (2025). Experiments were conducted on A800 GPUs; the exact hardware allocation is reported in Appendix G. Detailed hyperparameters and other implementation configurations are provided in Appendix A.

C Experimental Details

Table 10 summarizes the training and evaluation data used in the three environments. ReTool uses the public RL training and validation sets released with the ReTool project Feng et al. (2025), OpenThinkImg uses the chart-reasoning RL and held-out test sets Su et al. (2025), and ALFWorld follows the standard environment split for training and held-out evaluation Shridhar et al. (2021).

D ETD Analysis Details

We evaluate Qwen3-8B on the ALFWorld TextWorld environment Shridhar et al. (2021). Each game is rolled out four times. The agent is served with vLLM ; decoding is greedy ($T = 0$, top-$p = 1.0$) with at most 256 new tokens per turn and a repetition penalty of 1.05, so every trajectory is reproducible from the seed. An episode terminates when the environment signals task completion or when the budget of $B = 20$ turns is exhausted.

E Independent Runs and Confidence Intervals

To assess the variability of the reported results, we conduct three independent runs for each configuration. For trained configurations, each run includes both training and evaluation. For the untrained base models, only evaluation is repeated under the same protocol. Across runs, we keep the datasets, model configurations, hyperparameters, and evaluation metrics unchanged. We report the mean over the three runs together with its two-sided 95% confidence interval (CI).

Table 12: **Independent repeats and 95% CIs for the controlled failure-context interventions on ALFWorld at $\rho = 0.25$.** Values are changes in success rate (percentage points); E_k, N_k, and D_k denote error insertion, token-matched neutral insertion, and failure deletion, respectively.

Intervention	k	Runs 1/2/3	Mean $\pm$ 95% CI
Error insertion (E_k)	1	$-4.7/-4.2/-4.9$	-4.6 ± 0.90
	2	$-5.7/-5.7/-5.4$	-5.6 ± 0.43
	3	$-8.2/-8.0/-8.0$	-8.1 ± 0.29
Neutral insertion (N_k)	1	$-0.9/-1.2/-0.7$	-0.9 ± 0.63
	2	$-3.8/-3.6/-3.6$	-3.7 ± 0.29
	3	$-5.6/-5.4/-5.6$	-5.5 ± 0.29
Failure deletion (D_k)	1	$+2.5/+2.7/+2.7$	$+2.6 \pm 0.29$
	2	$+20.0/+19.8/+20.3$	$+20.0 \pm 0.63$

E.1 Independent Repeats for the ALFWorld Main Experiment

Table 11 reports the independent repeats for the ALFWorld portion of the main experiment. Across both model scales, TFAS obtains the highest mean Overall Acc and Err Acc among the compared reward strategies.

E.2 Independent Repeats for the Controlled Context Interventions

We repeat the controlled ALFWorld interventions at $\rho = 0.25$ from Section 3.3 three times. Each repeat reruns every matched branch with an independent evaluation seed while keeping the evaluated tasks, anchor construction, intervention content, and remaining action budget fixed. Table 12 reports the change in success rate relative to the corresponding matched trajectory with an unmodified history.

F Qualitative Analysis

This section provides additional qualitative examples from the three evaluation environments. Figure 5, Figure 6, and Figure 7 show representative cases from ReTool, OpenThinkImg, and ALFWorld, respectively. Together, these cases illustrate how TFAS changes agent behavior across code/tool-use reasoning, visual reasoning, and embodied planning.

G Compute Resources

All reinforcement-learning experiments were conducted on NVIDIA A800 GPUs with Ubuntu 22.04, CUDA 12.6, bfloat16 training, and PyTorch Fully Sharded Data Parallel (FSDP) Zhao et al. (2023). The OpenThinkImg experiments use 8 A800 GPUs per run for 6 hours. The remaining experiments, including all ReTool and ALFWorld runs, use 4 A800 GPUs per run. A ReTool training run takes approximately 50 hours, while an ALFWorld training run takes approximately 45 hours under the reported configuration. The training budgets, batch sizes, number of responses per prompt, maximum prompt and response lengths, and optimization steps are reported in Tables 4–9.

H Limitations

The method introduces a mixing weight λ, which controls the strength of tool-feedback advantage relative to task-accuracy advantage. Although our sensitivity study shows that a moderate value



Figure 5: **Qualitative example on ReTool.** The case shows how TFAS affects tool-augmented mathematical reasoning behavior.

works well, this hyperparameter may still need to be tuned for different environments, models, or tool-feedback definitions. TFAS also adds a small amount of training-time computation because it estimates and normalizes a separate trajectory-level tool-feedback advantage before policy optimization. This overhead is minor compared with rollout generation and model updates in our experiments, but it is still an additional component beyond outcome-only RL.



Figure 6: **Qualitative example on OpenThinkImg.** The case shows how TFAS affects tool-augmented visual reasoning behavior.

I Broader Impact

This work aims to improve the reliability of long-horizon language-model agents by reducing invalid tool calls, execution errors, and other forms of failure-induced context pollution. Potential positive impacts include more dependable agent behavior in tool-assisted reasoning, visual analysis, and embodied planning settings, especially when repeated trial-and-error interactions would otherwise create avoidable failures.



Figure 7: **Qualitative example on ALFWorld.** The case shows how TFAS affects embodied planning behavior.

As with general methods for improving agent capabilities, downstream applications should still consider the reliability and oversight requirements of the target domain. Better tool-use behavior may reduce avoidable execution errors, but it does not by itself guarantee that an agent's goals, tool permissions, or deployment context are appropriate. Our experiments are limited to benchmark environments and do not introduce new user data, scraped datasets, deployed systems, or released high-risk models. We therefore view the direct societal risk of this work as limited, while recommending that practical deployments use task-appropriate safeguards such as tool permission controls and human review for consequential decisions.

J OTHER

We use the following existing assets and cite their original sources in the manuscript. All assets are used for research evaluation and training under their respective usage terms.

- **Benchmarks and datasets.** The experiments use ReTool Feng et al. (2025), OpenThinkImg Su et al. (2025), and ALFWorld Shridhar et al. (2021) as the main interactive environments. The corresponding training and evaluation sets are DAPO-Math-17k Yu et al. (2025) and AIME 2025 Mathematical Association of America (2025) for ReTool, OpenThinkIMG-Chart-RL-14501 Su et al. (2025) and CharXiv Wang et al. (2024) for OpenThinkImg, and the standard ALFWorld splits Shridhar et al. (2021) for embodied planning.
- **Models.** The trainable base models are Qwen3-4B-Instruct-2507, Qwen3-8B Yang et al. (2025), Qwen3-VL-4B-Instruct, and Qwen3-VL-8B-Instruct Bai et al. (2025). Additional closed-source models are used only for failure-conditioned evaluation in Section 3, following their public API terms.
- **Training and evaluation frameworks.** The reinforcement-learning pipeline is implemented with verl verl Project (2024) and PyTorch FSDP Zhao et al. (2023). Model loading and evaluation use standard open-source libraries, including Transformers Wolf et al. (2020).