SAND: Boosting LLM Agents with Self-Taught Action Deliberation

Yu Xia[1] **Yiran Shen**[1] **Junda Wu**[1] **Tong Yu**[2] **Sungchul Kim**[2]
Ryan A. Rossi[2] **Lina Yao**[3,4] **Julian McAuley**[1]
[1]University of California San Diego [2]Adobe Research
[3]University of New South Wales [4]CSIRO's Data61

Abstract

Large Language Model (LLM) agents are commonly tuned with supervised finetuning on ReAct-style expert trajectories or preference optimization over pairwise rollouts. Most of these methods focus on imitating specific expert behaviors or promoting chosen reasoning thoughts and actions over rejected ones. However, without reasoning and comparing over alternative actions, LLM agents finetuned with these methods may over-commit towards seemingly plausible but suboptimal actions due to limited action space exploration. To address this, in this paper we propose Self-taught ActioN Deliberation (SAND) framework, enabling LLM agents to explicitly deliberate over candidate actions before committing to one. To tackle the challenges of when and what to deliberate given large action space and step-level action evaluation, we incorporate self-consistency action sampling and execution-guided action critique to help synthesize step-wise action deliberation thoughts using the base model of the LLM agent. In an iterative manner, the deliberation trajectories are then used to finetune the LLM agent itself. Evaluating on two representative interactive agent tasks, SAND achieves an average 20% improvement over supervised finetuning on initial expert data and also outperforms state-of-the-art agent tuning approaches.

1 Introduction

Large language models (LLMs) have recently been cast as agents that read instructions, reason through intermediate thoughts, and execute actions interacting with external environments such as web navigation [12, 13, 36], embodied household tasks [17], or scientific experiments [21]. Early prompting-based methods such as ReAct [22, 25, 38] interleave chain-of-thoughts and actions, enabling the LLM to plan and gather new information in context. To obtain more reliable LLM agents, recent works apply supervised finetuning on expert ReAct-style trajectories [2, 3, 4, 20, 42], or directly optimize on agent trajectory preference pairs [15, 19, 33].

Although effective, these approaches imitate expert actions or simply rank chosen actions over rejected actions and expose the model to mostly the reference action and corresponding rationale at each decision point. Without effectively exploring the action space, the agent seldom learns explicitly why the chosen action wins over plausible alternatives. As a result, the finetuned LLM agent can over-commit to superficially reasonable yet suboptimal actions, a failure mode also observed in self-consistency studies of LLMs [9, 23, 26]. Such behavior also hurts the generalization performance of LLM agents to unfamiliar scenarios.

To address this, in this paper we aim to teach LLM agent to deliberate by first generating several candidate actions for the current state, evaluating and comparing their likely outcomes, and then commit only after this evaluation. We propose Self-taught ActioN Deliberation (SAND) framework

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Bridging Language, Agent, and World Models for Reasoning and Planning.

to instantiate this idea by teaching the LLM agent with the deliberation thoughts synthesized by the base version of itself. However, as the action space of LLM agent tasks is often large or even unbounded [10, 36], it is intractable to deliberate over all actions and also inefficient to deliberate at every single step. To further tackle the challenge of when and what to deliberate, we devise self-consistency action sampling along expert trajectories to sample uncertain candidate actions of LLM agent at non-trivial decision making steps. To provide more informative and grounded step-level evaluations for each sampled candidate action, we utilize executed rollouts of each action to guide the critique generation. The action critiques are utilized to synthesize an action deliberation thought using the base LLM, which augments the initial expert trajectory and constructs deliberation trajectories for iterative finetuning of the LLM agent. Experiments on two interactive tasks demonstrate the advantage of our methods compared with strong agent tuning baselines. In summary, we make the following contributions:

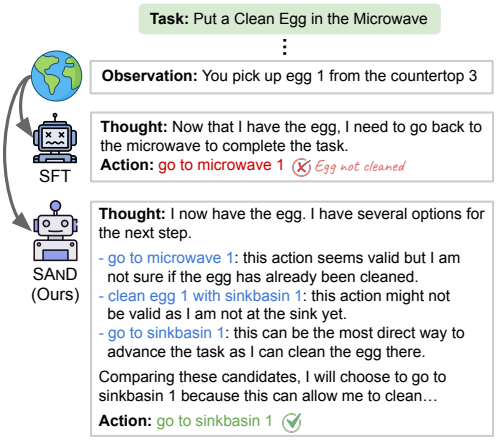

Figure 1: An illustrative example of an LLM agent task, where SFT trained agent [42] overcommits to a seemingly plausible but suboptimal action while our SAND tuned agent learns to deliberate over candidate actions before choosing the best action.

- To teach LLM agents better explore the action space, we propose Self-taught ActioN Deliberation (SAND), a self-learning framework teaching LLM agents to deliberatively reason over candidate actions before choosing one.
- To tackle the challenge of when and what to deliberate given large action space and step-level action evaluation, we devise self-consistency action sampling and execution-guided action critique to help synthesize high-quality deliberative reasoning thoughts for iterative finetuning.
- Experiments on two representative interactive agent tasks demonstrate the advantage of our method with an average 20% improvement over supervised finetuning on initial expert data and outperforming strong agent tuning baselines.

2 Related Work

2.1 LLM Agents Tuning

Recent efforts in tuning LLM agents have progressed from failure recovery heuristics towards more structured policy refinement. Early work such as FiReAct [2] showed that adding explicit failure-reflection demonstrations improves LLM agent robustness. AgentTuning [42] uses high-quality trajectories to finetune an instruction model for multi-turn interactions. ETO [19] retains exploratory trajectories and contrasts them with expert trajectories for agent optimization, while IPR [33] obtain step-level rewards for iterative preference refinement. DMPO [15] adapts direct preference optimization to multi-turn trajectory optimization, and WKM [14] regularizes actions with an external world knowledge model. Similarly, KnowAgent [45] teaches LLM agents for self action learning from a knowledge base [31] and NAT [20] incorporates failure trajectories for finetuning with an adapted prompt prefix. More recently, MPO [34] trains a meta planner agent that guides task execution agents. Several agent tuning benchmarks and datasets have also emerged [3, 18]. In contrast, our proposed SAND framework aims to teach LLM agents to effectively deliberate over candidate actions for better decision making.

2.2 Deliberative Reasoning

Prompting strategies for LLM deliberative reasoning have evolved rapidly. Chain-of-thought (CoT) prompting [24, 30] first showed that eliciting explicit intermediate reasoning steps markedly improves mathematical and symbolic reasoning. Building on this idea, ReAct blends CoT with environment feedback to couple reasoning and acting [38], while Self-Refine [11] and Reflexion [16] introduce

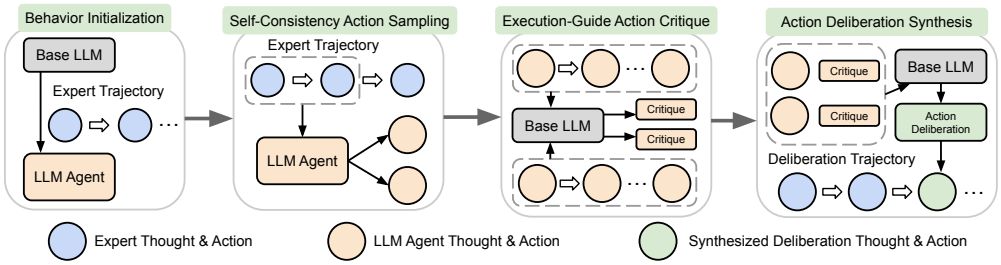

Figure 2: An illustration of our SAND framework for synthesizing one step of action deliberation.

iterative self-critique loops that rewrite faulty thoughts. Tree-of-Thought [37] generalizes CoT into a breadth-first search over alternative thought branches, allowing the model to back-track and globally evaluate solutions. SWAP [32] frames deliberate reasoning as structure-aware planning with an internal world model. Guan et al. [6] propose an explicit deliberation controller that decides when to generate, inspect or discard thoughts and Karanam et al. [8] study how many forward simulations are needed for reliable look-ahead in RL-style agents. Our SAnD framework extends the deliberative reasoning to LLM agent tasks with a focus of action deliberation.

2.3 Iterative Self Learning

Another relevant line of works enable a model to improve by repeatedly generating data and finetuning on its own synthesized output [29]. The idea began with STaR [41], which bootstraps a few verified solutions into a large corpus of correct rationales. RFT [40] generalises this to rejection-sampling proofs that pass an external checker. Subsequent work replaces hard filtering with self-feedback, e.g., Self-Refine [11] and SELF [2] alternate draft–critique–revise loops. Agent-R [39] repairs failed trajectories via Monte-Carlo search before re-training, and Karanam et al. [8] show that only a handful of such self-play iterations are needed before returns saturate. Our SAND framework follows similar iterative self-learning idea to steadily improve LLM agents without additional human supervision.

3 Task Formulation

We formulate our studied agent tasks as multi-turn interactions between an LLM agent and a text-based environment following Song et al. [19] and Xiong et al. [34]. Specifically, for a ReAct-style [38] LLM agent, the task begins with an instruction $u \in \mathcal{U}$. At each step, the LLM agent generates a reasoning thought $z \in \mathcal{Z}$ and an action $a \in \mathcal{A}$. The environment then returns an observation $o \in \mathcal{O}$. At time step t, for an LLM agent π_θ with the past interaction history up to time step $t-1$ denoted as $h_{t-1} = (u, z_1, a_1, o_1, \ldots, o_{t-1})$, the reasoning thought is sampled conditioned on the interaction history $z_t \sim \pi_\theta(\cdot \mid h_{t-1})$ followed by the action $a_t \sim \pi_\theta(\cdot \mid h_{t-1}, z_t)$. Therefore, for a complete agent trajectory with L steps $e = (u, z_1, a_1, o_1, \ldots, o_{L-1}, z_L, a_L)$, the probability of generating is

$$\pi_\theta(e \mid u) \;=\; \prod_{t=1}^{L} \pi_\theta\big(z_t, a_t \mid h_{t-1}\big). \tag{1}$$

After the task episode terminates upon success or maximum steps, the environment returns a task score $r(u, e) \in [0, 1]$ as the task successful rate.

4 Methodology

In this section, we describe in details our proposed Self-taught ActioN Deliberation (SAND) framework. Starting from a base LLM, SAND iteratively finetunes it to be a stronger LLM agent using the deliberation thoughts generated by the base version of itself. An intuitive illustration of our framework for generating a single step of deliberation thoughts can be found in Figure 2. A more comprehensive overview of the entire iterative self-learning pipeline are presented in Algorithm 1.

Algorithm 1: Self-Taught Action Deliberation (SAND)

Input: $\mathcal{D}_{\text{exp}} = \{(u, z_1, a_1, o_1, \ldots, o_{L-1}, z_L, a_L)^{(i)}\}$: expert trajectories, I: number of self-taught iterations, N: number of sampled actions, π_{base}: base LLM, $\pi_\theta = \pi_{\text{base}}$: trainable LLM.

Output: Final LLM agent π_θ

Finetune π_θ on $\mathcal{D}_{\text{exp}}$: $\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{e\sim\mathcal{D}_{\text{exp}}}\big[\log \pi_\theta(e \mid u)\big]$

for $k = 1$ **to** I **do**

  $\pi_k \leftarrow \pi_\theta$, $\mathcal{D}_{\text{delib}} \leftarrow \emptyset$

  foreach $e = (u, z_1, a_1, o_1, \ldots, z_L, a_L) \in \mathcal{D}_{exp}$ **do**

    Initialize history $h_0 \leftarrow u$ and self-taught deliberation trajectory $\tilde{e} = (u)$

    for $t = 1$ **to** L **do**

      Sample N actions: $\{\hat{z}_t^{(n)}, \hat{a}_t^{(n)}\}_{n=1}^{N} \sim \pi_k(\cdot \mid h_{t-1})$

      if $|\{\hat{a}_t^{(1)}, \ldots, \hat{a}_t^{(N)}, a_t\}| = 1$ **then continue**

      Rollout each action: $\{\hat{e}_t, r_t\} \sim \pi_k(\cdot \mid h_{t-1}, \hat{z}_t, \hat{a}_t)$

      Generate critique for each action: $c_t \sim \pi_{\text{base}}(\cdot \mid \hat{a}_t, \hat{e}_t, r_t, \texttt{Prompt}_c)$,

      Synthesize action deliberation thought: $\tilde{z}_t \sim \pi_{\text{base}}(\cdot \mid \{(\hat{a}_t^{(n)}, c_t^{(n)})\}_{n=1}^{N+1}, \texttt{Prompt}_d)$

      $\tilde{e} \leftarrow \tilde{e} \cup (\tilde{z}_t, a_t, o_t)$; $h_t \leftarrow (h_{t-1}, z_t, a_t, o_t)$

    $\mathcal{D}_{\text{delib}} \leftarrow \mathcal{D}_{\text{delib}} \cup \{\tilde{e}\}$

  Finetune π_θ on $\mathcal{D}_{\text{delib}}$: $\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{\tilde{e}\sim\mathcal{D}_{\text{delib}}}\big[\log \pi_\theta(\tilde{e} \mid u)\big]$

  Set $\mathcal{D}_{\text{exp}} \leftarrow \mathcal{D}_{\text{delib}}$ for the next iteration

return π_θ

4.1 Behavior Initialization

We start from a base instruction-tuned LLM π_{base}. Following Song et al. [19] and Xiong et al. [33], we initialize an LLM agent with the basic reasoning and action behavior for completing the task via supervised finetuning (SFT) on a set of ReAct-style expert trajectories on training tasks $\mathcal{D}_{\text{exp}} = \{(u, e)^{(i)}\}_{i=1}^{|\mathcal{D}|}$ with the loss

$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{e\sim\mathcal{D}_{\text{exp}}}\big[\log \pi_\theta(e \mid u)\big]. \tag{2}$$

We then obtain the initial LLM agent policy π_θ for the subsequent iterative improvement.

4.2 Self-Consistency Action Sampling

With an LLM agent policy π_θ, we aim to further teach agent the action deliberation behavior. Two central questions here are (i) when the agent should invest extra thinking over actions and (ii) what actions to think about, especially within a large or even unbounded action space. To address them, we utilize self-consistency action sampling which offers a natural solution.

For each expert trajectory e, we replay every expert interaction and branch at each step t. Specifically, given expert interaction history h_{t-1}, the current policy π_θ samples N actions

$$\{\hat{a}_t^{(1)}, \ldots, \hat{a}_t^{(N)}\} \sim \pi_\theta(\cdot \mid h_{t-1}), \tag{3}$$

where we omit the sampled reasoning thoughts $\hat{z}_t$ here for notation simplicity. Together with the original expert action a_t, we form a candidate action set of size $N + 1$.

We then define an inconsistency indicator that flags whether deliberation is needed for step t:

$$\mathbf{1}_{\text{delib}}(t) = \mathbf{1}\Big(\big|\{\hat{a}_t^{(1)}, \ldots, \hat{a}_t^{(N)}, a_t\}\big| > 1\Big). \tag{4}$$

If all actions in the set are the same, $\mathbf{1}_{\text{delib}}(t) = 0$, showing that the predictive distribution $\pi_\theta(\cdot \mid h_{t-1})$ is sharply peaked, this suggests that the model is confident in conducting the expert action a_t or the decision at the current state is trivial. In this case, no extra reasoning or deliberation is needed. When the set contains more than one unique action, $\mathbf{1}_{\text{delib}}(t) = 1$, this suggests the uncertainty of the LLM agent at the current state, and generating an explicit deliberation thought can help the agent better choose among candidate actions.

Moreover, since every branch starts from a step on the expert trajectory e, the sampled actions $\hat{a}_t$ remain close to both the demonstration distribution and the current LLM policy distribution while still exploring diverse futures, thereby avoiding random exploration over the large action space.

4.3 Execution-Guided Action Critique

If the inconsistency indicator flags for action deliberation at step t, $\mathbf{1}_{\text{delib}}(t) = 1$, then next question is how LLM agent can learn to generate meaningful step-level action evaluations when deliberating over the candidate set. In typical multi-turn interaction tasks, the reward is often delayed till task completion [27, 44]. Therefore, to provide additional context and evaluation signals for each candidate action, we collect its full rollout by executing each action $\hat{e}_t \sim \pi_\theta(\cdot \mid h_{t-1}, \hat{a}_t)$ and obtain the final task reward $r_t \in [0, 1]$ from the training environment.

Then, for each candidate action rollout, we prompt the frozen base LLM to generate a verbal critique c_t of the candidate action $\hat{a}_t$ guided by its execution results $\hat{e}_t$ and r_t

$$c_t \sim \pi_{\text{base}}\big(\cdot \mid \hat{a}_t, \hat{e}_t, r_t, \texttt{Prompt}_c\big), \tag{5}$$

where $\texttt{Prompt}_c$ is the critique prompt detailed in Figure 5. It shows the action, the ensuing sequence of observations, and the final reward, and asks for a concise verdict that states whether the action advanced, hindered, or had no effect on task success. As the critique is verbalized natural language, we also specify in the prompt for the base LLM to record reusable commonsense knowledge (e.g., "eggs are more likely to be stored in the refrigerators") that is not tied to the specific task instance. Such commonsense snippets accumulate across rollouts and provide transferable cues for more informative step-level action evaluation than numerical values aggregated from Monte Carlo rollouts [10, 33].

4.4 Action Deliberation Synthesis

After all critiques $c_t^{(n)}$ on candidate actions $\hat{a}_t^{(n)}$ have been gathered, we prompt the base LLM π_{base} to generate a single deliberation thought. The prompt, detailed in Figure 6, instructs the LLM to first propose and analyze each candidate action explicitly, then compare over them, and give a rationale for the final action choice of the expert action a_t at the current step

$$\tilde{z}_t \sim \pi_{\text{base}}\big(\cdot \mid \{(\hat{a}_t^{(n)}, c_t^{(n)})\}_{n=1}^{N+1}, \texttt{Prompt}_d\big). \tag{6}$$

We then append $(\tilde{z}_t, a_t, o_t)$ to the self-augmented deliberation trajectory $\tilde{e}$ collected along each step and update the running history h_t.

Note that we keep the expert action a_t as the ground-truth action here assuming it is the optimal one at the current step. However, as some expert data is annotated by human or another LLM, the LLM agent being finetuned may explore better paths than the expert path [19, 33]. Thus, we also devise an optional expert switch mechanism that replaces the original expert action with a better explored action if the LLM agent finds a better rollout during execution in Section 4.3.

4.5 Iterative Deliberation Finetuning

Exploring through all training tasks, the collection of self-taught action deliberation trajectories is denoted by $\mathcal{D}_{\text{delib}} = \{(u, \tilde{e})^{(i)}\}_{i=1}^{|\mathcal{D}_{\text{delib}}|}$. We update the LLM agent π_θ with via the similar supervised finetuning objective

$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{\tilde{e} \sim \mathcal{D}_{\text{delib}}}\big[\log \pi_\theta(\tilde{e} \mid u)\big]. \tag{7}$$

Compared with the initial expert trajectories, the synthesized deliberation trajectories provide richer guidance on enabling the action deliberation behavior as well as on why an action is chosen among alternative candidates, rather than only what action to mimic. Moreover, as the action deliberation is synthesized only when the action inconsistency indicator t, $\mathbf{1}_{\text{delib}}(t) = 1$ defined in Equation 4.2 flags, the trajectories $\mathcal{D}_{\text{delib}}$ we collected are mixed with deliberation and non-deliberation steps. This also teaches the LLM agent when to conduct action deliberation, as justified by our empirical analysis discussed in Section 6.4. Note that the LLM agent finetuned on the deliberation trajectories does not perform any action sampling during inference time. Instead, it generates the entire action deliberation thought in one pass, as illustrated in Figure 1.

Finally, we set $\mathcal{D}_{\text{exp}} \leftarrow \mathcal{D}_{\text{delib}}$ and repeat the sampling, critique, synthesis, and finetuning loop for I iterations, steadily improving LLM agents with a base version of itself without additional human labels or annotations.

5 Experimental Setup

5.1 Datasets and Evaluation

We evaluate our proposed SAND agent tuning framework mainly in two representative interactive environments **ALFWorld** and **ScienceWorld** following Xiong et al. [34]. ALFWorld [17] provides a text-based household task environment that for natural language understanding and embodied reasoning. It provides only binary rewards of task success upon completion or termination. ScienceWorld [21] presents a text-based environment where agents perform elementary-level scientific experiments. It offers a granular reward system that quantifies partial progress toward scientific task goals. Both datasets include training sets and test sets for both seen and unseen tasks as reported in Table 1, allowing us to assess how well LLM agents finetuned with SAND can generalize to unseen scenarios. We also report additional evaluation results on a real-world web navigation task WebShop [36] in Appendix A. Following Song et al. [19] and Xiong et al. [33], we use the **Average Reward** across test tasks as our main evaluation metric. We set the decoding temperature to 0 for all agents when evaluating on the test sets to facilitate reproducibility.

Dataset	Train	Test Seen	Test Unseen	Action Space
ScienceWorld	1483	194	211	19
ALFWorld	3321	140	134	13

Table 1: Statistics of ALFWorld and SciWorld.

5.2 Baselines and Variants

We compare SAND with the following agent tuning baselines and variants

- **AgentTuning** [42]: a direct supervised finetuning approach on expert trajectories.
- **ETO** [19]: a representative agent tuning method leveraging an adapted direct preference optimization objective for contrastive agent trajectory pairs.
- **KnowAgent** [45]: a recent framework employing an additional action knowledge base for self learning of LLM agents.
- **WKM** [14]: an agent tuning method with a jointly optimized world knowledge model available during test time.
- **MPO** [34]: an optimization approach via training a meta planner agent generating explicit guidance for task execution agents.
- **SAND**$_{\text{w/o SAS}}$: a variant of our method which does not conduct self-consistency action sampling (SAS) but instead directly prompts the base LLM to generate N alternative candidate actions in context during action deliberation synthesis.
- **SAND**$_{\text{w/o EAC}}$: a variant of our method which skips the execution-guided action critique (EAC) stage and directly synthesize action deliberation thought with N sampled candidate actions.

For more comprehensive comparison, we also report results of prompting-based ReAct-style LLM agent based on proprietary and open-sourced models **GPT4o** [1] and **Llama-3.1-70B-Instruct** [5] collected by Xiong et al. [34], where an in-context example is given for all prompting-based models. We provide in Appendix B additional discussion and comparisons of our SAND framework with recent test-time search methods guided by process reward or Q-value models [10, 28, 43].

5.3 Implementation Details

We adopt two backbone models Llama-3.1-8B-Instruct [5] and Qwen2.5-7B-Instruct [35] as the base models and finetune them with our SAND framework. The initial expert trajectories are collected by Song et al. [19]. For behavior initialization step, we follow Song et al. [19] to set batch size of 64 with a learning rate of 1e-5 and a cosine scheduler for 3 epochs. At self-consistency action sampling step, the decoding temperature of the LLM agent π_θ is set to 1.0 for sampling $N = 5$ candidate actions as well as the subsequent rollout execution. The execution-guided action critique is generated by the base LLM π_{base} with the decoding temperature 0. Both prompts for critique

Model	Single Agent	ScienceWorld		ALFWorld		Average
		Seen	Unseen	Seen	Unseen	
Agents w/o Training						
GPT-4o [1]	✓	60.0	56.0	78.6	83.6	69.6
GPT-4o-mini [1]	✓	49.1	42.7	32.1	41.0	41.2
Llama-3.1-8B-Instruct [5]	✓	47.7	42.2	22.9	28.4	35.3
Llama-3.1-8B-Instruct + MPO [34]	✗	56.5	55.5	50.0	52.2	53.6
Qwen2.5-7B-Instruct [35]	✓	38.5	38.8	71.4	75.4	56.0
Llama-3.1-70B-Instruct [5]	✓	72.6	70.2	78.6	73.9	73.8
Llama-3.1-70B-Instruct + MPO [34]	✗	80.4	<u>79.5</u>	85.7	86.6	83.1
Agents w/ Training						
Qwen2.5-7B-Instruct + SFT [42]	✓	69.2	60.8	72.1	75.4	69.4
Llama-3.1-8B-Instruct + SFT [42]	✓	75.6	65.1	79.3	71.6	72.9
Llama-3.1-8B-Instruct + ETO [19]	✓	81.3	74.1	77.1	76.4	77.2
Llama-3.1-8B-Instruct + KnowAgent [45]	✓	81.7	69.6	80.0	74.9	76.6
Llama-3.1-8B-Instruct + WKM [14]	✗	82.1	76.5	77.1	78.2	78.5
Llama-3.1-8B-Instruct + ETO&MPO [34]	✗	83.4	**80.8**	85.0	79.1	82.1
Qwen2.5-7B-Instruct + SAND (Iteration 1)	✓	80.9	67.2	85.7	85.0	79.7
Qwen2.5-7B-Instruct + SAND (Iteration 2)	✓	83.2	69.9	85.0	89.6	81.9
Qwen2.5-7B-Instruct + SAND (Iteration 3)	✓	84.0	69.0	90.7	94.8	84.6
Llama-3.1-8B-Instruct + SAND (Iteration 1)	✓	<u>86.6</u>	77.5	<u>92.9</u>	91.8	86.0
Llama-3.1-8B-Instruct + SAND (Iteration 2)	✓	**88.7**	78.2	**94.3**	<u>94.0</u>	<u>88.8</u>
Llama-3.1-8B-Instruct + SAND (Iteration 3)	✓	85.7	79.1	**94.3**	**96.3**	**88.9**

Table 2: Average rewards of all compared methods on two datasets. SAND significantly improves LLM agents across different model backbones, outperforming proprietary LLMs as well as state-of-the-art multi-agent approaches.

generation and action deliberation synthesis are provided in Appendix C. We disable the expert action switch mechanism discussed in Section 4.4 on ScienceWorld as we empirically observe that some of the tasks have short-cuts that might boost LLM agents on training set but hurt performances on test set. For deliberation finetuning steps, we set similarly batch size of 64 and learning rate of 1e-5 for $I = 3$ iterations. To avoid overfitting, we train 3 epochs only for the first iteration of SAND and 1 epoch for later iterations. We use OpenRLHF [7] to implement our training framework and all experiments run on 8 NVIDIA A100 80GB GPUs.

6 Results

6.1 How does SAND perform compared with other agent tuning methods?

We show the results of all compared methods on both seen and unseen test tasks in Table 2. From the results, we observe a clear advantage of SAND which outperforms all baselines on ALFWorld by a large margin. On ScienceWorld, SAND also shows competitive performances matching or surpassing state-of-the-art multi-agent approach. For both Llama-3.1-8B-Instruct and Qwen-2.5-7B-Instruct as the backbone LLMs, SAND (Iteration 3) achieves an average over 20% performance boost compared with SFT on initial expert data.

Besides, with our iterative deliberation finetuning, we also observe a steady performance improvement across different iterations of SAND, demonstrating the effectiveness of our self learning framework requiring no additional human labels. Another notable observation is that on later iterations of SAND, agents trained on both Llama-3.1-8B-Instruct and Qwen-2.5-7B-Instruct exhibit strong generalization capabilities on ALFWorld unseen tasks, achieving high rewards even than seen tasks. We attribute the performance gains on unseen tasks to the action deliberation behavior learned by LLM agents during SAND iterations. Such action deliberation behavior enables LLM agents to explicitly analyze unseen actions and environments before committing one instead of relying mostly on seen action patterns learned during training tasks.

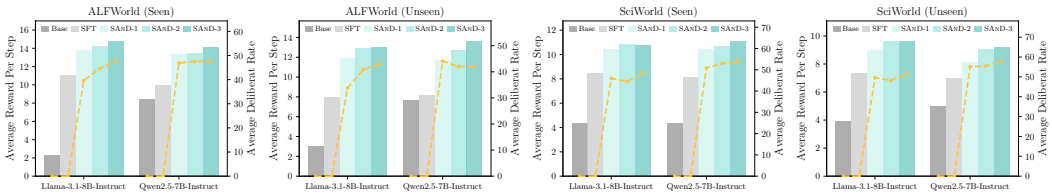

Figure 3: Average reward per step (bars) and average action deliberation rate per step (lines).

6.2 Are self-consistency action sampling and execution-guided critique necessary?

To validate the effectiveness of our devised self-consistency action sampling and execution-guided action critique, we compare SAND at the first iteration with its ablated variants $\text{SAND}_{\text{w/o SAS}}$ and $\text{SAND}_{\text{w/o EAC}}$. The results are shown in Table 3, where we observe a performance drop after removing each modules. Specifically, we find that $\text{SAND}_{\text{w/o SAS}}$ can even hurt the agent performance being outperformed by initial SFT. From our logged failed testing trajectories, we observe that without self-consistency action sampling, LLM agents often propose random actions irrelevant to the task goals and sometimes show degenerated behavior of repeating a candidate action till the maximum context length. On the other hand, $\text{SAND}_{\text{w/o EAC}}$, though also showing a small performance decrease compared with SAND, still improves over the initial SFT agent. The results again demonstrates the necessity of the self-consistency action sampling module while also validating the effectiveness of execution-guided action critique in improving the synthesized deliberation quality.

Method	ScienceWorld		ALFWorld	
	Seen	Unseen	Seen	Unseen
Qwen2.5-7B-Instruct				
Base	38.5	38.8	71.4	75.4
SFT	69.2	60.8	72.1	75.4
$\text{SAND}_{\text{w/o SAS}}$	63.5	52.4	72.1	62.7
$\text{SAND}_{\text{w/o EAC}}$	72.0	66.3	70.6	75.0
SAND	**80.9**	**67.2**	**85.7**	**85.0**
Llama-3.1-8B-Instruct				
Base	47.7	42.2	22.9	28.4
SFT	75.6	65.1	79.3	71.6
$\text{SAND}_{\text{w/o SAS}}$	70.3	62.0	85.7	77.3
$\text{SAND}_{\text{w/o EAC}}$	78.6	73.7	85.0	86.6
SAND	**86.6**	**77.5**	**92.9**	**91.8**

Table 3: Ablation study on modules in SAND.

6.3 Does action deliberation improve LLM agents at step-level across iterations?

SAND has shown overall performance improvement over iterations in Table 2. To further study the influence of the action deliberation behavior LLM agents learned from SAND, we show in Figure 3 the average reward per step and the corresponding average action deliberation rate per step across all test sets. The per-step average reward is calculated as the ratio of final reward to the total steps for each task, averaged across all tasks in the test set. Similarly, the per-step average deliberation rate is the ratio of action deliberation steps to the total steps for each task, averaged across all tasks.

From Figure 3, we can first observe a consistent improvement on per-step average reward across different finetuning iterations with first iteration shows a larger gain followed by smaller gains in later iterations. We also observe that the per-step action deliberation rate also show a general increasing pattern. Such correlation further validates the advantage of step-level action deliberation, which enables LLM agent to make better decisions at each step. The higher step-level reward also brings the advantages of earlier and more efficient task completion for practical applications of LLM agents.

6.4 Do LLM agents finetuned with SAND really learn when to deliberate?

To further analyze the agent tuning dynamics during SAND iterations, we study whether LLM agents have learned to decide when to deliberate over candidate actions, as discussed in Section 4.2. Specifically, we visualize when the LLM agent decides to deliberate with violin plots in Figure 4, where each panel corresponds to an iteration in SAND. As ScienceWorld provides finegrained rewards that can reflect partial task completion rate, we partition the unseen tasks on ScienceWorld into three difficulty bands based on the empirical tertiles of reward distribution from the base LLM Llama-3.1-8B-Instruct. We define the bottom third as *Hard* tasks, the middle third as *Medium* tasks, and the top third as *Easy* tasks. Within each band we compute the deliberation rate of SAND similarly

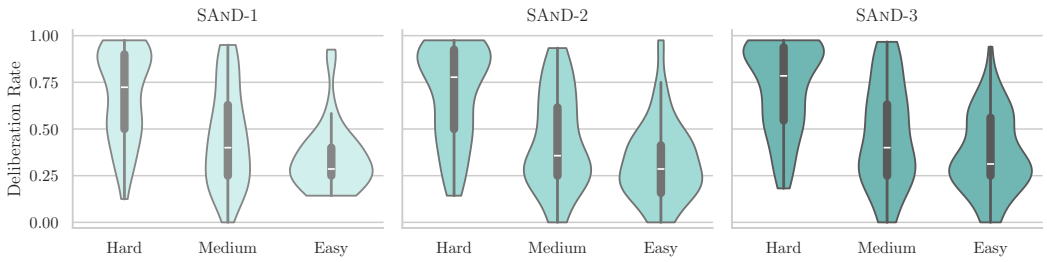

Figure 4: Action deliberation rate distribution across three difficulty bands in unseen test set on ScienceWorld. Each panel corresponds to a SAND iteration starting from Llama-3.1-8B-Instruct. The difficulty bands *Hard*, *Medium*, *Easy* are determined based on the tertiles of reward distribution from the base Llama-3.1-8B-Instruct. The results show that more SAND iterations teach LLM agents to deliberate more on hard tasks and less on easy tasks.

defined as the ratio of deliberation steps to the total steps for each task, and plot the distribution of deliberation rates across tasks.

From Figure 4, we observe that across all three iterations the hard band remains the only one with a high median deliberation rate around 0.75, while the median deliberation rate on easy band stays near 0.30. This shows SAND effectively teaches LLM agent to direct more action deliberation to hard tasks while keeping reasoning concise when the task is easy. From iteration 1 to iteration 3, we also observe a slight distribution shift of the hard violin, which widens at the top with the median gradually increases. This further demonstrates the effectiveness of iterative deliberation finetuning in our SAND framework that not only improves the task performances but also teaches LLM agents to make better decisions on when to deliberate.

6.5 How much additional inference-time computation cost does SAND introduce?

As SAND teaches LLM agents to explicitly deliberate over candidate actions, it introduces additional computation cost during inference time. To study how much additional inference-time cost is incurred, we compare in Table 4 the average number of tokens used per task between the SFT agent (without action deliberation) and our SAND-finetuned agents (with action deliberation), where the base model is Llama-3.1-8B-Instruct.

From Table 4, we find that the additional action deliberation results in approximately 2 to 3 times more tokens per task. Compared to representative test-time scaling approaches such as Best-of-N, which incurs 5 times more tokens when $N = 5$, we believe our SAND framework introduces a reasonable additional inference-time computation cost with considerable performance improvements. Moreover, as analyzed in Section 6.4, our SAND framework effectively teaches LLM agents when to deliberate, avoiding unnecessary action deliberation on simple tasks. This finding is also reflected in Table 4, where a slight decreasing trend in token usage is observed across iterations, indicating better inference-time computation usage through our iterative finetuning framework.

Method	ALFWorld	ScienceWorld
SFT	498.3	800.0
SAND (Iteration 1)	1,314.2 (2.6×)	2,411.9 (3.0×)
SAND (Iteration 2)	1,105.8 (2.2×)	2,522.1 (3.2×)
SAND (Iteration 3)	1,146.2 (2.3×)	2,253.6 (2.8×)

Table 4: Average #tokens per task on ALFWorld and SciWorld. Multipliers are relative to SFT.

7 Conclusion

In this paper, we propose Self-taught Action Deliberation (SAND), a self-learning framework that equips LLM agents with explicit action deliberation. Addressing when and what to deliberate given large action space, SAND samples candidate actions by self-consistency, critiques each action guided exectured rollout, synthesizes a deliberation thought, and iteratively finetunes the LLM agent on the enriched trajectories. Experiments and analysis demonstrate the effectivenes and advantages of our methods, which further highlights the key role of deliberative reasoning in developing more powerful LLM agents for real world applications.

Limitations

Despite the performance improvements, generating more deliberation thoughts inevitably increases the token usage and inference costs. As discussed and analyzed in Section 6.4, our proposed SAND framework teaches LLM agent when to deliberate via self-consistency action sampling to avoid deliberating during trivial decision making steps. Our results in Section 6.5 further show that the action deliberation learned by SAND introduces reasonable additional inference-time computation cost. To further improve the reasoning efficiency, more advanced methods such reinforcement learning or direct preference optimization can be utilized to guide the LLM agent to better decide when to generating more comprehensive deliberative reasoning and when to generate more concise quick thoughts. Parallel inference can also be applied to further enhance the inference efficiency.

References

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

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

[3] Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. Agent-flan: Designing data and methods of effective agent tuning for large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 9354–9366, 2024.

[4] Zhixun Chen, Ming Li, Yuxuan Huang, Yali Du, Meng Fang, and Tianyi Zhou. Atlas: Agent tuning via learning critical steps. *arXiv preprint arXiv:2503.02197*, 2025.

[5] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[6] Melody Y Guan, Manas Joglekar, Eric Wallace, Saachi Jain, Boaz Barak, Alec Helyar, Rachel Dias, Andrea Vallone, Hongyu Ren, Jason Wei, et al. Deliberative alignment: Reasoning enables safer language models. *arXiv preprint arXiv:2412.16339*, 2024.

[7] Jian Hu, Xibin Wu, Zilin Zhu, Xianyu, Weixun Wang, Dehao Zhang, and Yu Cao. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*, 2024.

[8] Arjun Karanam, Farnaz Jahanbakhsh, and Sanmi Koyejo. Towards deliberating agents: Evaluating the ability of large language models to deliberate. In *NeurIPS 2024 Workshop on Behavioral Machine Learning*, 2024.

[9] Xun Liang, Shichao Song, Zifan Zheng, Hanyu Wang, Qingchen Yu, Xunkai Li, Rong-Hua Li, Yi Wang, Zhonghao Wang, Feiyu Xiong, et al. Internal consistency and self-feedback in large language models: A survey. *arXiv preprint arXiv:2407.14507*, 2024.

[10] Zongyu Lin, Yao Tang, Xingcheng Yao, Da Yin, Ziniu Hu, Yizhou Sun, and Kai-Wei Chang. Qlass: Boosting language agent inference via q-guided stepwise search. *arXiv preprint arXiv:2502.02584*, 2025.

[11] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.

[12] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

[13] Dang Nguyen, Jian Chen, Yu Wang, Gang Wu, Namyong Park, Zhengmian Hu, Hanjia Lyu, Junda Wu, Ryan Aponte, Yu Xia, Xintong Li, Jing Shi, Hongjie Chen, Viet Dac Lai, Zhouhang Xie, Sungchul Kim, Ruiyi Zhang, Tong Yu, Mehrab Tanjim, Nesreen K. Ahmed, Puneet Mathur, Seunghyun Yoon, Lina Yao, Branislav Kveton, Jihyung Kil, Thien Huu Nguyen, Trung Bui, Tianyi Zhou, Ryan A. Rossi, and Franck Dernoncourt. GUI agents: A survey. In Wanxiang Che,

Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 22522–22538, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1158. URL https://aclanthology.org/2025.findings-acl.1158/.

[14] Shuofei Qiao, Runnan Fang, Ningyu Zhang, Yuqi Zhu, Xiang Chen, Shumin Deng, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Agent planning with world knowledge model. *Advances in Neural Information Processing Systems*, 37:114843–114871, 2024.

[15] Wentao Shi, Mengqi Yuan, Junkang Wu, Qifan Wang, and Fuli Feng. Direct multi-turn preference optimization for language agents. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2312–2324, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.138. URL https://aclanthology.org/2024.emnlp-main.138/.

[16] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

[17] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

[18] Yifan Song, Weimin Xiong, Xiutian Zhao, Dawei Zhu, Wenhao Wu, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. Agentbank: Towards generalized llm agents via fine-tuning on 50000+ interaction trajectories. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 2124–2141, 2024.

[19] Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization of LLM agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7584–7600, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.409. URL https://aclanthology.org/2024.acl-long.409/.

[20] Renxi Wang, Xudong Han, Yixuan Zhang, Timothy Baldwin, and Haonan Li. NAT: Enhancing agent tuning with negative samples. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7385–7398, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. URL https://aclanthology.org/2025.naacl-long.378/.

[21] Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? *arXiv preprint arXiv:2203.07540*, 2022.

[22] Ruoyu Wang, Junda Wu, Yu Xia, Tong Yu, Ryan A Rossi, Julian McAuley, and Lina Yao. Dice: Dynamic in-context example selection in llm agents via efficient knowledge transfer. *arXiv preprint arXiv:2507.23554*, 2025.

[23] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

[24] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[25] Junda Wu, Yu Xia, Tong Yu, Xiang Chen, Sai Sree Harsha, Akash V Maharaj, Ruiyi Zhang, Victor Bursztyn, Sungchul Kim, Ryan A. Rossi, Julian McAuley, Yunyao Li, and Ritwik Sinha. Doc-react: Multi-page heterogeneous document question-answering. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 67–78, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-252-7. doi: 10.18653/v1/2025.acl-short.6. URL https://aclanthology.org/2025.acl-short.6/.

[26] Yu Xia, Xu Liu, Tong Yu, Sungchul Kim, Ryan Rossi, Anup Rao, Tung Mai, and Shuai Li. Hallucination diversity-aware active learning for text summarization. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8665–8677, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.479. URL https://aclanthology.org/2024.naacl-long.479/.

[27] Yu Xia, Tong Yu, Zhankui He, Handong Zhao, Julian McAuley, and Shuai Li. Aligning as debiasing: Causality-aware alignment via reinforcement learning with interventional feedback. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4684–4695, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.262. URL https://aclanthology.org/2024.naacl-long.262/.

[28] Yu Xia, Jingru Fan, Weize Chen, Siyu Yan, Xin Cong, Zhong Zhang, Yaxi Lu, Yankai Lin, Zhiyuan Liu, and Maosong Sun. Agentrm: Enhancing agent generalization with reward modeling. *arXiv preprint arXiv:2502.18407*, 2025.

[29] Yu Xia, Subhojyoti Mukherjee, Zhouhang Xie, Junda Wu, Xintong Li, Ryan Aponte, Hanjia Lyu, Joe Barrow, Hongjie Chen, Franck Dernoncourt, Branislav Kveton, Tong Yu, Ruiyi Zhang, Jiuxiang Gu, Nesreen K. Ahmed, Yu Wang, Xiang Chen, Hanieh Deilamsalehy, Sungchul Kim, Zhengmian Hu, Yue Zhao, Nedim Lipka, Seunghyun Yoon, Ting-Hao Kenneth Huang, Zichao Wang, Puneet Mathur, Soumyabrata Pal, Koyel Mukherjee, Zhehao Zhang, Namyong Park, Thien Huu Nguyen, Jiebo Luo, Ryan A. Rossi, and Julian McAuley. From selection to generation: A survey of LLM-based active learning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14552–14569, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.708. URL https://aclanthology.org/2025.acl-long.708/.

[30] Yu Xia, Rui Wang, Xu Liu, Mingyan Li, Tong Yu, Xiang Chen, Julian McAuley, and Shuai Li. Beyond chain-of-thought: A survey of chain-of-X paradigms for LLMs. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert, editors, *Proceedings of the 31st International Conference on Computational Linguistics*, pages 10795–10809, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.719/.

[31] Yu Xia, Junda Wu, Sungchul Kim, Tong Yu, Ryan A. Rossi, Haoliang Wang, and Julian McAuley. Knowledge-aware query expansion with large language models for textual and relational retrieval. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4275–4286, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. URL https://aclanthology.org/2025.naacl-long.216/.

[32] Siheng Xiong, Ali Payani, Yuan Yang, and Faramarz Fekri. Deliberate reasoning for llms as structure-aware planning with accurate world model. *arXiv preprint arXiv:2410.03136*, 2024.

[33] Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. Watch every step! LLM agent learning via iterative step-level process refinement. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1556–1572, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.93. URL https://aclanthology.org/2024.emnlp-main.93/.

[34] Weimin Xiong, Yifan Song, Qingxiu Dong, Bingchan Zhao, Feifan Song, Xun Wang, and Sujian Li. Mpo: Boosting llm agents with meta plan optimization. *arXiv preprint arXiv:2503.02682*, 2025.

[35] 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.

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

[37] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

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

[39] Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. Agent-r: Training language model agents to reflect via iterative self-training. *arXiv preprint arXiv:2501.11425*, 2025.

[40] Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023.

[41] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

[42] Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. AgentTuning: Enabling generalized agent abilities for LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3053–3077, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.181. URL https://aclanthology.org/2024.findings-acl.181/.

[43] Yuanzhao Zhai, Tingkai Yang, Kele Xu, Dawei Feng, Cheng Yang, Bo Ding, and Huaimin Wang. Enhancing decision-making for llm agents via step-level q-value models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 27161–27169, 2025.

[44] Chen Zhang, Xinyi Dai, Yaxiong Wu, Qu Yang, Yasheng Wang, Ruiming Tang, and Yong Liu. A survey on multi-turn interaction capabilities of large language models. *arXiv preprint arXiv:2501.09959*, 2025.

[45] Yuqi Zhu, Shuofei Qiao, Yixin Ou, Shumin Deng, Shiwei Lyu, Yue Shen, Lei Liang, Jinjie Gu, Huajun Chen, and Ningyu Zhang. KnowAgent: Knowledge-augmented planning for LLM-based agents. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 3709–3732, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. URL https://aclanthology.org/2025.findings-naacl.205/.

Appendix

A Additional Results on Webshop

To further verify the generalizability of SAND to more diverse environments, we report in Table 5 the performance of SAND with Llama-3.1-8B-Instruct as the base model on a real-world web navigation task WebShop [36]. We use the same train-test dataset splits as in Song et al. [19]. The number of sample actions in our self-consistency action sampling is set to $N = 3$ due to the smaller action space of WebShop compared to ALFWorld and SciWorld. Other configurations remain the same as in Section 5.3. From the results, we observe a consistent performance boost with our SAND framework for LLM agents with around 10% improvement compared to the SFT baseline, which validates the effectiveness of SAND on more diverse environments.

Method	WebShop
Base	55.3
SFT	65.4
SAND (Iteration 1)	68.5
SAND (Iteration 2)	72.4
SAND (Iteration 3)	71.8

Table 5: Average rewards on WebShop.

B Comparisons with PRM and Q-Value Models for LLM Agents

In this work, we propose an LLM agent tuning framework, SAND, that enhances LLM agents' abilities during training time with self-taught deliberation trajectories. During inference, our SAND-finetuned LLM agent generates the entire action deliberation thought along with the final action in one pass, as illustrated in Figure 1. Therefore, our proposed LLM agent tuning framework is orthogonal and complementary to recent process reward model (PRM) or Q-value model-guided test-time search methods [10, 28, 43], which train separate reward or value models and perform multiple samplings at each step during inference.

Though our method is compatible with those test-time search techniques for LLM agents, for a more comprehensive view, we report in Table 6 some preliminary comparisons of SAND with representative test-time search methods guided by PRMs [28] and Q-value models [10, 43]. Note that the results are directly imported from the original papers and thus the base models might be slightly different. We leave further integration of our agent tuning framework with advanced test-time search methods as future work.

Method	Train Base LLM Agent	Train Separate PRM/Value Model	Inference-time Sampling Strategy	WebShop	ALFWorld (Unseen)	SciWorld (Unseen)
Llama-3.1-8B-Instruct + Q [43]	✗	✓	5 Actions Per Step	60.0	–	–
Llama-2-7B-Chat + QLASS [10]	✓	✓	6 Actions Per Step	70.3	82.8	66.4
Llama-3-8B-Instruct + AgentRM-BoN [28]	✓	✓	Best-of-5 Trajectories	71.0	94.8	76.1
Llama-3-8B-Instruct + AgentRM-Beam [28]	✓	✓	25 Actions Per Step (5×5 Beam Search)	75.3	96.3	82.6
Llama-3.1-8B-Instruct + SAND (Ours)	✓	✗	1 Action Per Step (No Sampling)	72.4	96.3	79.1

Table 6: Comparisons of SAND with representative test-time search methods guided by PRM or Q-value model.

C Prompts

In this section, we provide the prompts used in our SAND framework. The prompt for execution-guided critique generation is shown in Figure 5 and prompt for action deliberation synthesis is shown in Figure 6. For evaluation on test set of ALFWorld and ScienceWorld, we follow the same prompts used in Xiong et al. [34] for fair comparison, which is provided in Figure 7 and Figure 8.

Prompt for Execution-Guided Action Critique

```
### Background
{task_instruction}

### Current State
{interaction_history}

### Private Mental Simulations
You quietly imagined several futures that all start with the action
    **{sample_action}**.
Here is your simulated futures (keep it private):

{executed_rollout}

### Instructions
Write **one short paragraph (3 sentences)** titled exactly
`Action Evaluation:` that captures **your** intuitive judgement of
executing **{sampled_action}** now. In fluent prose, incorporate any of the
    following aspects as you see fit:

* Whether **{sampled_action}** in the current state is valid based on the
    environment feedback.
* Whether and how it might help advance the current progress toward important
    sub-goals or final goal of completing the task.
* Any task-relevant affordances or commonsense cues you should notice.
* Frequent failure patterns or error loop you should be cautious for similar
    tasks.
* A practical evaluation of the action **{sampled_action}** in the current state.

Do **not** directly quote or refer to the simulation log, and do **not** list
    items; blend them naturally into the paragraph.
Do **not** mention that the simulations exists or that you had outside help.

### Output Format
Action Evaluation: <your paragraph>
```

Figure 5: Prompt used for the execution-guided action critique.

Prompt for Action Deliberation Synthesis

```
Background
{task_instrution}

### Current State
{interaction_history}

### Private Scratch-pad
You silently drafted several possible next actions with your intuitive judgement
    about each (these notes stay private):
- {candidate_action_1}: {critique_for_candidate_action_1}
- {candidate_action_2}: {critique_for_candidate_action_2}
- {candidate_action_3}: {critique_for_candidate_action_3}

### Very Important
Your final **Action** line must be **{expert_action}**. Everything you write has
    to lead naturally to this choice.

### Instructions
Generate reasoning thoughts following the instructions below:
Begin with a short one-sentence reflection of your previous action and your
    current situation.
Then propose and list each candidate action from the scratch-pad with your own
    intuitive judgement, e.g., - <candidate action>: <your judgement>.
Keep your judgement informative and avoid repeating generic evaluation
    statements.

Do **not** mention that the scratch-pad exists or that you got outside help.

### Output Format
Thought: <your one-sentence reflection>

- <candidate action>: <your judgement>
- <candidate action>: <your judgement>

<your comparison and rationale>
```

Figure 6: Prompt used for action deliberation synthesis.

Prompt for ALFWorld Tasks

```
Interact with a household to solve a task. Imagine you are an intelligent agent
    in a household environment and your target is to perform actions to complete
    the task goal. At the beginning of your interactions, you will be given the
    detailed description of the current environment and your goal to accomplish.
For each of your turn, you will be given the observation of the last turn. You
    should choose from two actions: "Thought" or "Action". If you choose
    "Thought", you should first think about the current condition and plan for
    your future actions, and then output your action in this turn. Your output
    must strictly follow this format:"Thought: your thoughts.\n Action: your
    next action"; If you choose "Action", you should directly output the action
    in this turn. Your output must strictly follow this format:"Action: your
    next action".
The available actions are:
1. go to {recep}
2. take {obj} from {recep}
3. put {obj} in/on {recep}
4. open {recep}
5. close {recep}
6. toggle {obj} {recep}
7. clean {obj} with {recep}
8. heat {obj} with {recep}
9. cool {obj} with {recep}
where {obj} and {recep} correspond to objects and receptacles.
After your each turn, the environment will give you immediate feedback based on
    which you plan your next few steps. if the envrionment output "Nothing
    happened", that means the previous action is invalid and you should try more
    options.
Reminder:
1. The action must be chosen from the given available actions. Any actions
    except provided available actions will be regarded as illegal.
2. Think when necessary, try to act directly more in the process.

Now, it's your turn and here is the task.
{task}
```

Figure 7: Prompt used for ALFWorld tasks.

Prompt for ScienceWorld Tasks

```
You are a helpful assistant to do some scientific experiment in an environment.
In the environment, there are several rooms: kitchen, foundry, workshop,
    bathroom, outside, living room, bedroom, greenhouse, art studio, hallway
You should explore the environment and find the items you need to complete the
    experiment.
You can teleport to any room in one step.
All containers in the environment have already been opened, you can directly get
    items from the containers.
For each of your turn, you will be given the observation of the last turn. You
    should choose from two actions: "Thought" or "Action". If you choose
    "Thought", you should first think about the current condition and plan for
    your future actions, and then output your action in this turn. Your output
    must strictly follow this format:"Thought: your thoughts.\n Action: your
    next action"; If you choose "Action", you should directly output the action
    in this turn. Your output must strictly follow this format:"Action: your
    next action". Remember that you can only output one "Action:" in per
    response.

The available actions are:
open OBJ: open a container
close OBJ: close a container
activate OBJ: activate a device
deactivate OBJ: deactivate a device
connect OBJ to OBJ: connect electrical components
disconnect OBJ: disconnect electrical components
use OBJ [on OBJ]: use a device/item
look around: describe the current room
examine OBJ: describe an object in detail
look at OBJ: describe a container's contents
read OBJ: read a note or book
move OBJ to OBJ: move an object to a container
pick up OBJ: move an object to the inventory
pour OBJ into OBJ: pour a liquid into a container
mix OBJ: chemically mix a container
teleport to LOC: teleport to a specific room
focus on OBJ: signal intent on a task object
wait: task no action for 10 steps
wait1: task no action for a step

Now, it's your turn and here is the task.
{task}
```

Figure 8: Prompt used for ScienceWorld tasks.