DEEPPLANNER: Scaling Planning Capability for Deep Research Agents via Advantage Shaping

Anonymous ACL submission

Abstract

Large language models (LLMs) augmented with multi-step reasoning and action generation abilities have shown promise in leveraging external tools to tackle complex tasks that require long-horizon planning. However, existing approaches either rely on implicit planning in the reasoning stage or introduce explicit planners without systematically addressing how to optimize the planning stage. As evidence, we observe that under vanilla reinforcement learning (RL), planning tokens exhibit significantly higher entropy than other action tokens, revealing uncertain decision points that remain under-optimized. To address this, we propose **DEEPPLANNER**, an end-to-end RL framework that effectively enhances the planning capabilities of deep research agents. Our approach shapes token-level advantage with an entropy-based term to allocate larger updates to high entropy tokens, and selectively upweights sample-level advantages for planning-intensive rollouts. Extensive experiments across seven deep research benchmarks demonstrate that DEEPPLANNER improves planning quality and achieves state-of-the-art results under a substantially lower training budget.

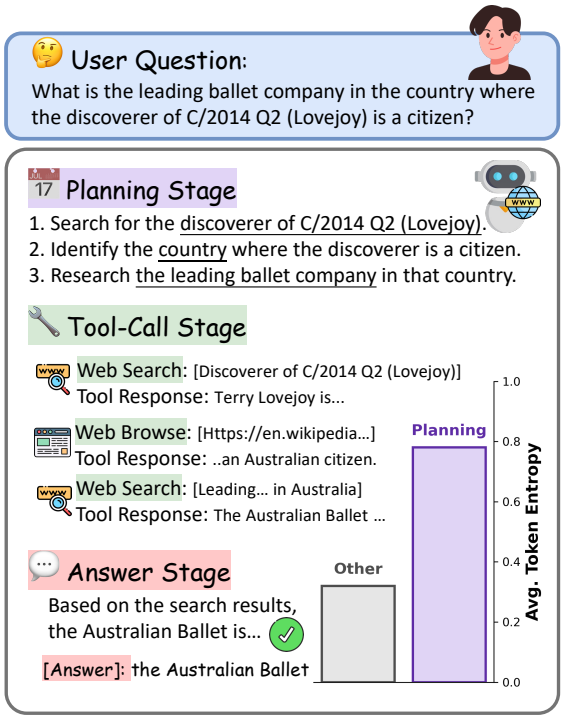

Figure 1: When the agent is instructed to explicitly write a research plan and then execute it, we observe that the planning stage exhibits much higher token entropy (0.78) than other stages (0.32) during RL training.

1 Introduction

The evolution of AI assistants has progressed from simple question-answering systems to sophisticated research agents capable of complex information synthesis. While early systems relied solely on the internal knowledge of large language models (LLMs), the introduction of retrieval-augmented generation (RAG) (Lewis et al., 2021) greatly expanded their knowledge by incorporating external documents. Deep research agents(OpenAI, 2025a; Gemini, 2025; xAI, 2025; Zheng et al., 2025) represent the next leap in this evolution, combining advanced reasoning capabilities (DeepSeek-AI et al., 2025; Jaech et al., 2024) with action generation (Yao et al., 2023) to orchestrate diverse external tools (*e.g.*, web search). These agents automate the complete research workflow of discovering, verifying, and summarizing online information (Xu and Peng, 2025), enabling them to tackle research tasks that traditionally require human expertise.

However, simple reactive approaches that alternate between thinking and acting suffer from inefficiency and goal drift in long-horizon research tasks, necessitating explicit high-level planning to decompose complex goals and coordinate multi-stage workflows (Huang et al., 2024; Wei et al., 2025a; Xu and Peng, 2025). While some systems like DeepResearcher (Zheng et al., 2025) operate without upfront planning, allowing plans to emerge implicitly during reasoning, others, such as `OpenAI/Agents SDK`(OpenAI, 2025b) and Cogni-

tive Kernel-Pro(Fang et al., 2025), incorporate dedicated planner components for goal decomposition and execution tracking. Despite these framework advances, it still lacks systematic approaches to diagnose and optimize the planning capabilities that are crucial for deep research agents' success.

To diagnose this gap, we extend DeepResearcher (Zheng et al., 2025) into a plan-then-execute framework that explicitly decouples high-level planning from low-level execution (*e.g.*, tool calls and answer generation). The agent must propose an initial plan in the first round within **`<plan> ...</plan>`** and can refine it as new evidence arrives. Under vanilla Group Relative Policy Optimization (GRPO) (Shao et al., 2024), we uncover a consistent pattern shown in Figures 1 and 3: planning tokens exhibit substantially higher entropy than other execution tokens throughout training. Coupled with the performance–entropy transformation mechanism (Cui et al., 2025), this observation reveals the challenge that existing methods cannot effectively convert elevated planning-stage entropy into improved downstream performance.

To address this, we introduce DEEPPLANNER, an end-to-end reinforcement learning (RL) framework with advantage shaping that sustainably scales the planning capability of deep research agents. First, inspired by (Cheng et al., 2025), we append an entropy-shaped term to the original token-level advantages, amplifying gradients on uncertain tokens (primarily during planning) while clipping to prevent sign flips on strongly negative advantages. This detached shaping term primarily reinforces advantageous planning trajectories and prevents entropy collapse, preserving sustained exploration. Second, to further strengthen performance on planning-intensive tasks, we introduce selective advantage upweighting. Prior work (Zhang et al., 2025) filters rollouts with more tool calls within each RL iteration, then performs supervised fine-tuning (SFT) on these trajectories before continuing RL training. Instead, within each rollout group under the same query, we identify the most efficient rollout (correct answer, fewest tool calls) and define its tool-call count as the query complexity. We then upweight sample-level advantages for the most efficient rollout in groups exceeding the complexity threshold, achieving comparable gains while maintaining end-to-end simplicity.

Extensive experiments demonstrate that DEEPPLANNER achieves state-of-the-art (SOTA) results on deep research benchmarks with markedly fewer training resources: 3,072 queries and 8 rollouts per query, versus 10× more training samples and roughly 2× more rollouts of the previous SOTA framework EvolveSearch (Zhang et al., 2025). Ablations further show that (i) explicit planning improves performance on long-horizon tasks, (ii) entropy-based advantage shaping accelerates effective planning optimization without entropy collapse, and (iii) selective advantage upweighting better exploits complex rollouts that require intensive planning.

Our contributions are summarized as follows:

- We diagnose and quantify persistent high entropy on planning tokens within a plan-then-execute deep research framework, revealing untapped potential to scale planning capacity.
- We propose DEEPPLANNER, which introduces two advantage shaping mechanisms under GRPO to amplify learning on uncertain planning tokens and on complex, high-quality rollouts, enabling efficient end-to-end training without interleaved SFT.
- We conduct extensive experiments demonstrating that DEEPPLANNER achieves SOTA performance with markedly reduced budgets. Ablations further characterize planning–entropy dynamics and quantify how advantage shaping scales planning capability.

2 DEEPPLANNER: Preliminaries

In this section, we detail the definition of deep research, define the trajectory structure, and describe the basic modules and tools that constitute the environment in which our agent operates.

2.1 Problem Definition

In the deep research scenario, given a user query q and system prompt p, the LLM-based agent follows the ReAct (Yao et al., 2023) framework to perform multi-turn reasoning (*a.k.a.* thinking) and action, including *plan*, *tool call*, and *answer*, until producing a final output. The overall process can be represented as a trajectory:

$$\tau = \{(s_0, e_0, a_0), (s_1, e_1, a_1), \ldots, (s_T, e_T, a_T), R\}, \quad (1)$$

where s_t denotes the state at step t, consisting of the system prompt p, user query q, accumulated reasoning (*a.k.a.* thinking) tokens $\{e_i\}_{i=0}^{t}$, action tokens $\{a_i\}_{i=0}^{t}$, and tool responses up to t. At each

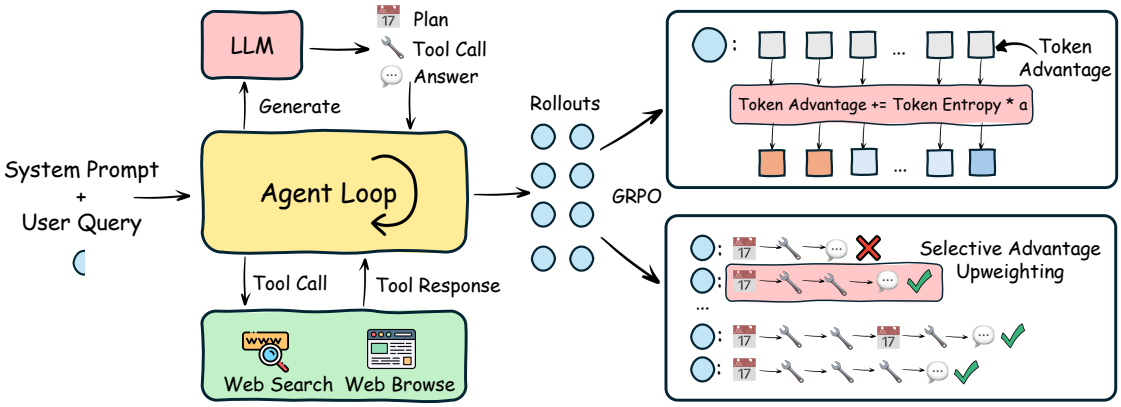

Figure 2: The overview of DEEPPLANNER. For each rollout, GRPO token-level advantages are augmented with $\alpha\times$ token entropy (clipped to avoid sign flips). Within each group, rollout(s) that reach the correct answer with fewer tool calls receive a higher weight (tool-call counts gate complexity).

step t, the model generates (i) a thinking segment e_t, and (ii) an action token a_t, which correspond to a high-level plan or a low-level execution. R denotes the terminal reward that evaluates the quality of the final answer with respect to the user query.

2.2 Agent Loop Modules

At each step, the output of our deep research agent can be decomposed into the following modules:

Think. Before taking any explicit action, we instruct the agent to think and generate a reasoning segment, wrapped in `<think>...</think>`, serving as the model's latent observation and analysis about the current state s_t.

Plan. The plan is the only high-level action in the trajectory. Unlike prior approaches where planning is often entangled with reasoning or execution, we require the model to explicitly output its intended strategy within **`<plan>...</plan>`** tags. The plan specifies what information to search, what tools to invoke, and how to integrate the tool response for answering the query. Two constraints are imposed: (1) an initial plan must be provided in the first round, and (2) the plan can be refined or revisited in later steps. This explicit separation yields interpretable, verifiable plans and discourages opportunistic, ad hoc strategies that can destabilize long-horizon reasoning.

Tool Call. As a low-level action, a tool call can invoke either `web_search` or `web_browse` external tool. The agent produces a structured request, following a pre-specified tool schema as detailed in Appendix B.1, and encloses it in `<tool_call>...</tool_call>`. The environment then executes the request, returning structured tool outputs as the tool responses, which are appended to the agent context for the next step.

Answer. The answer module denotes the low-level termination action. When the agent determines that sufficient information has been gathered, it outputs the final response in `<answer>...</answer>`. Executing this action ends the trajectory, and the agent receives the terminal reward R, which reflects answer quality.

2.3 Deep Research Tools

During the deep research pipeline, our environment provides two external tools (Zheng et al., 2025):

Web Search. The web search tool is invoked when the agent requires external evidence sources. The agent issues a JSON-formatted request specifying the tool name `web_search` and query string(s). The tool calls the online web search API (Serper[1]) and returns a ranked list of top-k results (we set $k = 10$), each containing a *title, URL, and snippet* as follows:

```
Request: [query1, query2, ...]
Response: (title1, URL1, snippet1), (...
```

Furthermore, both the search query and returned results are cached for downstream use by the web browsing tool.

Web Browse. When a deeper inspection of candidate URLs is needed, the agent calls the browsing tool. The request consists of a target URL together with the associated user query q and search query retrieved from the cache. The browsing agent first fetches the initial page segment of the document, summarizes query-relevant content, and stores it

[1] https://serper.dev/

in short-term memory. Based on this memory, it adaptively decides whether to continue reading subsequent segments of the URL or to stop. Once browsing concludes, the accumulated short-term memory is compiled and returned as the browsing tool response as follows:

```
Request: (query, [URL1, URL2, URL3...])
Response: (URL1, page information1), (...
```

Without returning the entire webpage content, the browsing tool enables selective reading and iterative evidence extraction, which reduces context length pressure on the agent while maintaining information relevance.

3 DEEPPLANNER: Advantage Shaping

In this section, we provide a detailed introduction to our end-to-end training framework based on GRPO (Shao et al., 2024), with our proposed advantage shaping methodology, which enables more efficient planning optimization and high-quality rollout utilization.

3.1 Vanilla RL Training Framework

We train the policy π_θ of DEEPPLANNER using reinforcement learning under the GRPO framework.

GRPO Unlike traditional sample-level loss functions (DeepSeek-AI et al., 2025), GRPO in this paper operates at the token level, which is particularly important for long text generation tasks as sample-level optimization tends to dilute the contribution of tokens in longer sequences within a group. Formally, the objective function is given by:

$$\mathcal{J}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|x)} \frac{1}{\sum_{i=1}^{G}|y_i|} \sum_{i=1}^{G}\sum_{t=1}^{|y_i|} \Big[\min\big(r_{i,t}A_{i,t},\ \text{clip}(\cdot)A_{i,t}\big) - \beta D_{\text{KL}}\Big]. \tag{2}$$

Here, G denotes the number of rollouts τ generated by the agent for the same prompt. A rollout y refers only to the tokens generated by the agent, while τ additionally includes tool responses. $|y_i|$ represents the length of the i-th rollout. The importance sampling ratio is defined as $r_{i,t}(\theta) = \frac{\pi_\theta(a_{i,j,t}|s,a_{i,j,<t})}{\pi_{\text{old}}(a_{i,j,t}|s,a_{i,j,<t})}$. D_{KL} measures the discrepancy between the current policy π_θ and the reference policy π_{old}. The clipping function is defined as $\text{clip}(1-\epsilon, r_{i,t}, 1+\epsilon)$, where ϵ and β are hyperparameters. Finally, the group relative advantage is computed as $A_{i,j} = \frac{r_i - \text{mean}(r)}{\text{std}(r)}$, which are computed using rollout rewards within the same group.

Rewards. The reward function is designed to balance correctness with adherence to the required output structure: (1) **Format Reward**: $+0.5$ if the output strictly follows the required structure (`<plan>`, `<tool_call>`, `<answer>`). (2) **Answer Reward**: $+0.5$ if the final answer matches the ground truth. In this work, we employ an LLM-as-a-judge approach to determine the answer reward, as detailed in Appendix B.3. Importantly, format violations override correctness. For example, if the output format is incorrect (e.g., invalid tool call syntax or failing to generate a `<plan>` in the first round), the reward is set to 0, even if the final answer is correct.

3.2 Entropy-based Advantage Shaping (EAS)

Under vanilla GRPO, as shown in Figure 3, planning tokens (`<think>` and `<plan>` segments in the planning steps) remain consistently higher entropy than other stages throughout training, indicating that the agent retains substantial uncertainty when forming plans. Empirically, policy performance is fundamentally traded from policy entropy (Cui et al., 2025), making it critical to guide planning entropy toward a stable, reasonable range during training to improve downstream performance. However, unlike RL tasks in mathematics (Shao et al., 2024) or coding, deep research training faces severe constraints due to long tool response times, making it impractical to scale training over hundreds of steps. Consequently, high-entropy planning receives insufficient advantage signals for optimization, leaving substantial room for exploration unexploited. On the other hand, simply increasing the learning rate can accelerate convergence, but it also leads to *entropy collapse*, where the model quickly converges to rigid patterns with minimal diversity and ceases to improve. Inspired by (Cheng et al., 2025), to accelerate the optimization of the planning capability and encourage exploration while preventing collapse, we add a gradient-detached entropy-based shaping term to the original token advantages:

$$\begin{aligned} \psi(\mathcal{H}_{i,t}) &= \min\left(\alpha \cdot \mathcal{H}_{i,t}^{\text{detach}},\ \frac{|A_{i,t}|}{\kappa}\right), \\ A_{i,t}^{\text{EAS}} &= A_{i,t} + \psi(\mathcal{H}_{i,t}), \end{aligned} \tag{3}$$

where $A_{i,t}$ is the original token advantage, $\mathcal{H}_{i,t}$ is the token-level entropy (detached from the computation graph), α is a tunable shaping coefficient, and $\kappa > 1$ is a clipping factor that prevents negative advantages from being flipped positively, thereby preventing poor actions from being turned into favorable ones solely due to high entropy.

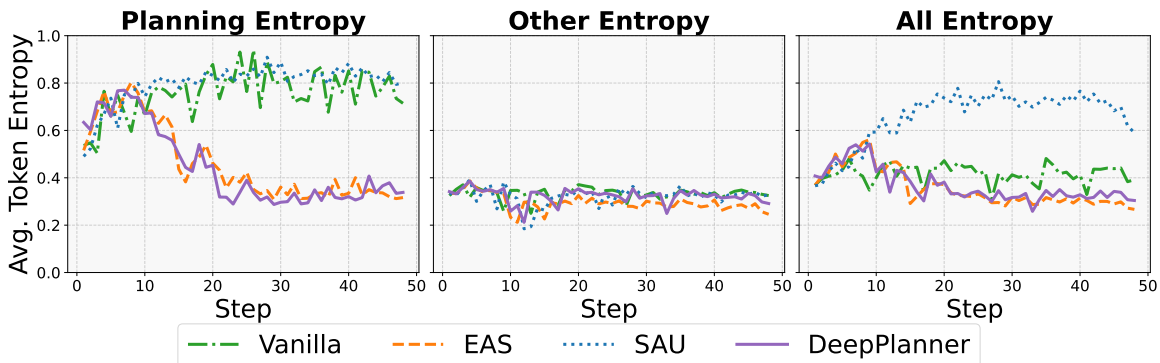

Figure 3: In the vanilla GRPO framework, planning-stage entropy is markedly higher than other execution-stage entropy, revealing underdeveloped planning capacity (Cui et al., 2025). With the advantage shaping mechanisms, our DEEPPLANNER effectively transforms the planning entropy to the performance on deep research tasks without inducing global entropy collapse.

3.3 Selective Advantage Upweighting (SAU)

Long-horizon tasks often contain multiple rollouts of varying difficulty. Rollouts that are more complex tend to contribute more significantly to model improvement. Prior work (Zheng et al., 2025) observes that the average number of tool calls increases gradually over the course of training. Building on this, (Zhang et al., 2025) proposed an iterative pipeline: filter rollouts that are both correct and belong to the top-k in tool-call usage, apply supervised fine-tuning (SFT) on them, and then resume RL training. While effective, this approach introduces considerable engineering complexity.

To achieve a similar and more efficient effect in an end-to-end manner, we introduce *selective advantage upweighting*, which amplifies the advantages of high-quality, complex rollouts during RL training. This serves as an additional shaping strategy that mimics the benefits of SFT filtering while maintaining a simpler training pipeline. Furthermore, we optimize the filtering strategy used for rollout selection. As illustrated in Figure 2, within each rollout group, we select trajectories that: (1) achieve the maximum reward (1.0), and (2) use the minimum number of tool calls N_{tool} in the group while exceeding a threshold c controlling complexity. Formally, we define:

$$A_{i,t}^{\text{SAU}} = A_{i,t} \cdot \lambda, \quad N_{tool} \geq c, \tag{4}$$

where $\lambda > 1$ controls the strength of upweighting. When both entropy-based and selective shaping are applied together in DEEPPLANNER, the update rule is:

$$A_{i,t}^{\text{Shaping}} = \begin{cases} A_{i,t} \cdot \lambda + \psi(\mathcal{H}_{i,t}), \text{if } \tau_i \text{ is selected} \\ A_{i,t} + \psi(\mathcal{H}_{i,t}), \text{if } \tau_i \text{ is not selected} \end{cases} \tag{5}$$

Compared with previous methods that select rollouts purely based on tool-call counts, our strategy not only enforces a lower bound on rollout complexity but also promotes more efficient strategies, rather than simply encouraging excessive tool calls. This avoids over-rewarding redundant tool usage and reduces wasted resources.

4 Experiments

4.1 Experiment Settings

Dataset and Metrics We follow the training and evaluation protocol of DeepResearcher. The training set comprises NQ, TQ, HotpotQA, and 2Wiki in a 1:1:3:3 ratio, totaling 80,000 samples, with 75% being multi-hop, matching the long-horizon information-seeking scenarios of deep research. For evaluation, as detailed in Appendix B.3, we adopt model-based evaluation (MBE) (Zhang et al., 2025) using `chatgpt-4o-latest` as the judge to score final answers (correct = 1, incorrect = 0), and we report the average MBE on (1) In-domain: NQ, TQ, HotpotQA, and 2Wiki (2,048 examples). (2) Out-of-domain: Musique, Bamboogle, and PopQA (1,129 examples). Note that our RL training uses only a small subset (3,072) of the entire training set, and Table 2 summarizes the training budgets for our method and the baselines.

Training Configuration In this paper, we choose `Qwen2.5-7B-Instruct` as the backbone model and the judge model during RL training. Each optimization step samples 64 queries and generates 8 rollouts for each query. For DEEPPLANNER and ablation studies, we all train for 48 steps. For EAS, we set the shaping coefficient $\alpha = 0.1$ and the clipping factor $\kappa = 2$, following previous

Method	Inference Environment	In-domain					Out-of-domain				Total
		NQ	TQ	Hotpot	2Wiki	Avg	Musique	Bamb	PopQA	Avg	
◇ ***Prompt Based***											
CoT	-	32.0	48.2	27.9	27.3	33.9	7.4	21.6	15.0	14.7	25.7
CoT + RAG	Local RAG	59.6	75.8	43.8	24.8	51.0	10.0	27.2	48.8	28.7	41.4
Search-o1*	Local RAG	57.4	61.1	40.8	32.8	48.0	21.3	38.4	42.4	34.0	42.0
Search-o1	Web Search	55.1	69.5	42.4	37.7	51.2	19.7	53.6	43.4	38.9	45.9
♣ ***Training Based***											
Search-r1-base	Local RAG	60.0	76.2	63.0	47.9	61.8	27.5	57.6	47.0	44.0	54.2
Search-r1-instruct	Local RAG	49.6	49.2	52.5	48.8	50.0	28.3	47.2	44.5	49.5	49.8
R1-Searcher	Web Search	52.3	79.1	53.1	65.8	62.6	25.6	65.6	43.4	44.9	55.0
DeepResearcher	Web Search	61.9	85.0	64.3	66.6	69.5	29.3	72.8	52.7	51.6	61.8
DeepResearcher*	Web Search	66.4	86.0	65.4	75.0	73.2	29.0	71.7	50.2	50.3	63.4
EvolveSearch-ite1	Web Search	68.5	87.4	65.4	75.6	74.2	29.3	74.0	51.2	51.5	64.5
EvolveSearch-ite2	Web Search	69.4	86.3	66.3	**78.5**	75.1	31.6	76.5	52.8	53.6	65.9
EvolveSearch-ite3	Web Search	71.0	**89.5**	67.7	76.4	**76.2**	33.8	77.1	50.3	53.7	66.6
♠ ***Our Series***											
DEEPPLANNER	Web Search	72.9	87.5	**68.2**	75.6	76.1	32.4	77.6	**55.3**	55.1	**67.1**
- w/ *EAS*	Web Search	73.4	85.9	66.6	72.1	74.5	**34.4**	**78.4**	54.5	**55.8**	66.5
- w/ *SAU*	Web Search	70.3	86.7	67.0	64.8	72.2	33.4	76.0	54.7	54.7	64.7
- *Vanilla*	Web Search	**73.8**	87.3	65.8	68.8	73.9	30.1	71.2	53.7	51.7	64.4

Table 1: The overall performance across seven benchmarks shows that DEEPPLANNER achieves SOTA results in terms of the total average MBE score and significantly enhances the out-of-domain generalization. We also report DeepResearcher with model-based reward (Zhang et al., 2025) (denoted DeepResearcher*)

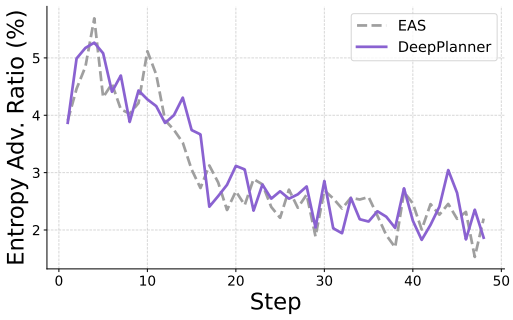

Figure 4: The ratio $\frac{\mathcal{H}_{i,t}}{|A_{i,t}|}$ indicates that our approach does not over-encourage during the training process.

Method	# Samples	# Rollouts
DeepResearcher	8,000	16
EvolveSearch-ite1	16,000	16
EvolveSearch-ite2	24,000	16
EvolveSearch-ite3	32,000	16
DEEPPLANNER	3,072	8

Table 2: Comparison of training sample size and rollout configurations across different methods.

work (Cheng et al., 2025). For SAU, we set $\lambda = 2$ and complexity threshold $c = 2$, as approximately 40% of rollouts involve at least two tool calls. Furthermore, we provide more implementation details in Appendix A. In the final evaluation, we report four configurations for ablation: (1) Vanilla: GRPO without advantage shaping. (2) w/ EAS: GRPO with entropy-based shaping only. (3) w/ SAU: GRPO with selective upweighting only. (4) DEEPPLANNER: GRPO with both entropy-based shaping and selective upweighting.

Baselines To evaluate the effectiveness of DEEPPLANNER, we compare against a series of baselines: (1) CoT Only (Wei et al., 2023): Chain-of-Thought prompting without external retrieval. (2) RAG (Gao et al., 2024): CoT with retrieved reference context for answer generation. (3) Search-o1 (Li et al., 2025a): A training-free baseline that sends search queries via APIs (e.g., Serper) and browses the webpage for richer evidence. (5) Search-r1 (base/instruct) (Jin et al., 2025): RL-based framework trained with Wikipedia retrieval at train and inference time, using *Qwen2.5-7B-base* and *Qwen2.5-7B-Instruct* as the backbone models. (6) R1-Searcher (Song et al., 2025): Uses Bing search and answers by summarizing the first three result pages. (7) DeepResearcher (Zheng et al., 2025): Open-web deep research with autonomous URL selection rather than fixed top-3 webpage summarization. (8) EvolveSearch (Zhang et al., 2025): An RL–SFT loop using model-based reward. After each RL round, it filters the top-2,000 rollouts with the highest tool-call counts for SFT, then continues to the next RL round (four RL rounds and three SFT rounds in total).

4.2 Overall Performance

With fewer training samples and rollouts per sample, DEEPPLANNER achieves the best overall MBE: 67.1 with 3,072 samples and 8 rollouts, surpassing EvolveSearch-ite3 trained on 32,000 samples and 16 rollouts (see Tables 1 and 2). This underscores that scaling high-level planning quality, rather than merely scaling data or rollouts, is critical for improving deep research performance. The entropy of planning token declines over training (Figure 3), reflecting growing plan confidence. The ratio between the entropy-based shaping term and original advantage (Figure 4) also drops, indicating our shaping avoids over-encouraging confident tokens, preventing premature entropy collapse while preserving exploration.

4.3 Detailed Analysis

Explicit planning matters. Analogous to how CoT elicits multi-step reasoning, explicitly requiring the agent to produce a global plan before acting yields structured, verifiable strategies. Compared with DeepResearcher∗ without explicit planning, enforcing the `<plan>` stage improves MBE from 63.4 to 64.4, confirming that making intentions explicit stabilizes long-horizon behavior.

Format accuracy first boosts performance, and planning sustains gains. As illustrated in Figure 7, the reward climbs rapidly around step 15, coinciding with a small entropy drop for tool-call and answer tokens, as well as a shift in the rollout distribution across reward tiers (reward < 0.5 indicates format error). The early performance surge is primarily due to better adherence to the required output structure. However, planning token entropy remains comparatively high at that point, and further reducing plan entropy correlates with sustained performance gains, which DEEPPLANNER and *Entropy Adv* ablation achieve via entropy-based advantage shaping.

Entropy-based shaping accelerates optimization while avoiding over-encouragement. Our entropy-based shaping amplifies learning signals on uncertain planning tokens, accelerating the discovery of effective strategies while clipping prevents flipping strongly negative advantages. Figures 5 and 6 indicate that the model learns concise, efficient plans under shaping. Unlike simply increasing the learning rate, our method preserves exploration and avoids entropy collapse (Figure 3), preventing rigid, brittle behaviors.

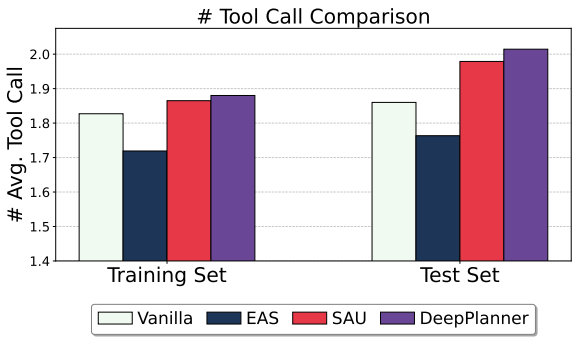

Figure 5: The average number of tool calls between different approaches in both the training and test sets.

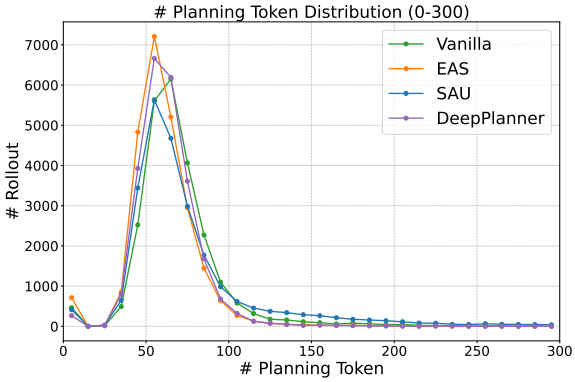

Figure 6: Distribution of planning tokens, with token length above 300 not reported.

Selective advantage upweighting encourages prudence by increasing tool calls. Selective advantage upweighting approximates the benefits of RL–SFT loops with minimal engineering effort. Our selective mechanism encourages efficient, not excessive, tool use. Figure 5 shows a measured increase in tool calls where warranted, boosting in-domain performance from 74.5 to 76.1 and improving total performance. SAU alone does increase entropy (Figure 3), reflecting increased uncertainty on harder problems and slower convergence. Nevertheless, it improves out-of-domain generalization (51.7 to 54.7), indicating robustness from targeted practice on complex cases.

EAS + SAU: faster learning on complex rollouts with stable plan convergence. Combining entropy-based shaping with selected rollouts focuses learning on the most impactful tokens within the most beneficial rollouts. The model simultaneously converges to reliable plans and learns to exercise prudence by allocating more tool calls on complex instances, yielding the best overall results (total 67.1).

Case Study As shown in Figure 8, we remove the explicit planning stage from DEEPPLANNER, retrain under identical settings, and observe a typical short-sighted failure. With planning, the agent proposes a two-step plan (identify the father → find his birthplace), makes two focused `web_search` calls, cross-checks sources, and returns the correct answer. Without planning, it only plans the first step, ignores global dependencies, mixes goals with missing intermediates, drifts via name concatenation (e.g., to Evan O'Neill Kane), wastes calls, and outputs the wrong answer, underscoring why explicit planning stabilizes entity resolution.

5 Related Work

5.1 Deep Research Agents

Early prompt-based search agents rely on fixed workflows to query and integrate external knowledge. Systems such as OpenResearcher (Zheng et al., 2024b), AirRAG (Feng et al., 2025), Iter-DRAG (Yue et al., 2025), Search-o1 (Li et al., 2025a), and Open Deep Search (Alzubi et al., 2025) enhance search capabilities through carefully crafted prompts and interaction patterns, but their hand-engineered designs limit adaptability and generalization. To overcome these limits, SFT-based approaches (Yu et al., 2024) learn more flexible retrieval and synthesis policies. For example, CoRAG (Wang et al., 2024) couples SFT with MCTS to dynamically select document blocks under budgets, trading off compute at planning time and sensitivity to supervised signals. RL-based methods utilize final outcome-supervised optimization to train end-to-end research agents, pushing the frontier of autonomous deep research capability, as demonstrated by ReSearch (Chen et al., 2025), R1-Searcher (Song et al., 2025), Search-R1 (Jin et al., 2025), WebRL (Qi et al., 2025), WebThinker (Li et al., 2025b), WebAgent-RL (Wei et al., 2025b), DeepResearcher (Zheng et al., 2025), and EvolveSearch (Zhang et al., 2025). Across these paradigms, planning is central: for complex problems, task decomposition typically helps LLMs execute more accurately and transparently. Plan*RAG (Verma et al., 2025) employs a separate model to produce an explicit plan and verifies its gains for information retrieval. Cognitive Kernel-Pro (Fang et al., 2025) introduces a planner that maintains a structured to-do list and a completed list, and DeepResearcher (Zheng et al., 2025) observes the emergence of planning ability during RL. In this paper, we provide the first systematic analysis of how planning influences RL-based deep research agents and introduce an end-to-end method that scales their planning abilities via advantage shaping (Cheng et al., 2025).

5.2 Planning Capability of LLMs

The planning capability of LLMs (Wei et al., 2025a) to decompose high-level goals into actionable, temporally coordinated steps emerges as a central component of agentic systems. Broadly, existing approaches fall into two paradigms: (1) prompting LLMs to produce plans directly, which are then executed or lightly post-processed by downstream systems (Wei et al., 2023; Wang et al., 2023; Qin et al., 2023; Liang et al., 2023; Ahn et al., 2022; Guo et al., 2023; Zheng et al., 2024a); and (2) using LLMs to draft intermediate plans that are subsequently verified, refined, or expanded by symbolic planners, specialized agents, or external tools (Liu et al., 2023; Singh et al., 2022; Yuan et al., 2023; Kambhampati et al., 2024; Li et al., 2025c). In research-intensive, open-ended settings, systems such as (OpenAI, 2025b; Fang et al., 2025) exemplify the value of explicit planning by separating plan generation from action execution, thereby enabling interpretable, verifiable, and progress-tracking behaviors. To further explore the planning ability in deep research agents, we provide a systematic, token-level entropy analysis that reveals persistently high plan-stage entropy as a key bottleneck, and introduce an advantage-shaping method to concentrate learning on uncertain planning decisions and complex rollouts, thereby directly scaling the agent planning capacity.

6 Conclusion

In this paper, we identify persistently high entropy in planning tokens of deep research agents, revealing untapped optimization potential. To address this, we propose DEEPPLANNER with two advantage shaping mechanisms: entropy-guided token-level shaping that accelerates planning optimization while preventing collapse, and selective up-weighting that prioritizes complex, high-quality rollouts. Our approach achieves state-of-the-art performance with lower training budgets, demonstrating that targeted advantage shaping effectively scales planning capability in deep research agents.

Limitations

First, due to the high cost of end-to-end RL in deep research tasks, we cap DEEPPLANNER and all ablations at 48 RL steps for fair comparison, requiring 24 hours on 8×A100 (80GB) GPUs, around $50 in LLM API spend (mostly webpage-browsing summaries) and $50 in Serper API for web search. Planning-token entropy remains relatively high, indicating headroom to further scale exploration of planning capability. Second, following EvolveSearch's analysis of model-based judgments, we do not independently evaluate the judge's reliability, and we directly adopt `chatgpt-4o-latest` as the evaluator for fair comparison with EvolveSearch (Zhang et al., 2025). Finally, our method focuses on advantage shaping without explicit plan-stage rewards. Alternative approaches using fine-grained, multi-dimensional process reward for planning (e.g., quality, feasibility, consistency, verifiability) represent a different research direction deserving exploration.

Ethics Statement

To the best of our knowledge, the backbone model (Team, 2024) and datasets (Zheng et al., 2025) used in this work are open-source and legally permissible for research use. Our experiments employ LLMs[2] and tools[3] under their respective licenses, and we adhered to the terms of service for all APIs used in this work.

[2] https://openrouter.ai/
[3] https://serper.dev/

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, and 26 others. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *Preprint*, arXiv:2204.01691.

Salaheddin Alzubi, Creston Brooks, Purva Chiniya, Edoardo Contente, Chiara von Gerlach, Lucas Irwin, Yihan Jiang, Arda Kaz, Windsor Nguyen, Sewoong Oh, Himanshu Tyagi, and Pramod Viswanath. 2025. Open deep search: Democratizing search with open-source reasoning agents. *Preprint*, arXiv:2503.20201.

Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. 2025. Research: Learning to reason with search for llms via reinforcement learning. *Preprint*, arXiv:2503.19470.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. 2025. Reasoning with exploration: An entropy perspective on reinforcement learning for llms. *Preprint*, arXiv:2506.14758.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. 2025. The entropy mechanism of reinforcement learning for reasoning language models. *Preprint*, arXiv:2505.22617.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Tianqing Fang, Zhisong Zhang, Xiaoyang Wang, Rui Wang, Can Qin, Yuxuan Wan, Jun-Yu Ma, Ce Zhang, Jiaqi Chen, Xiyun Li, Hongming Zhang, Haitao Mi, and Dong Yu. 2025. Cognitive kernel-pro: A framework for deep research agents and agent foundation models training. *Preprint*, arXiv:2508.00414.

Wenfeng Feng, Chuzhan Hao, Yuewei Zhang, Guochao Jiang, Jingyi Song, and Hao Wang. 2025. Airrag: Autonomous strategic planning and reasoning steer retrieval augmented generation. *Preprint*, arXiv:2501.10053.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv:2312.10997.

Gemini. 2025. Gemini deep research.

Yiduo Guo, Yaobo Liang, Chenfei Wu, Wenshan Wu, Dongyan Zhao, and Nan Duan. 2023. Learning to plan with natural language. *Preprint*, arXiv:2304.10464.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024. Understanding the planning of llm agents: A survey. *Preprint*, arXiv:2402.02716.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *Preprint*, arXiv:2503.09516.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. 2024. Llms can't plan, but can help planning in llm-modulo frameworks. *Preprint*, arXiv:2402.01817.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025a. Search-o1: Agentic search-enhanced large reasoning models. *Preprint*, arXiv:2501.05366.

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yutao Zhu, Yongkang Wu, Ji-Rong Wen, and Zhicheng Dou. 2025b. Webthinker: Empowering large reasoning models with deep research capability. *Preprint*, arXiv:2504.21776.

Ziyue Li, Yuan Chang, Gaihong Yu, and Xiaoqiu Le. 2025c. Hiplan: Hierarchical planning for llm-based agents with adaptive global-local guidance. *Preprint*, arXiv:2508.19076.

Yaobo Liang, Chenfei Wu, Ting Song, Wenshan Wu, Yan Xia, Yu Liu, Yang Ou, Shuai Lu, Lei Ji, Shaoguang Mao, Yun Wang, Linjun Shou, Ming Gong, and Nan Duan. 2023. Taskmatrix.ai: Completing tasks by connecting foundation models with millions of apis. *Preprint*, arXiv:2303.16434.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. 2023. Llm+p: Empowering large language models with optimal planning proficiency. *Preprint*, arXiv:2304.11477.

OpenAI. 2025a. Deep research system card.

OpenAI. 2025b. Openai agents sdk.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, Tianjie Zhang, Wei Xu, Jie Tang, and Yuxiao Dong. 2025. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *Preprint*, arXiv:2411.02337.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *Preprint*, arXiv:2307.16789.

ByteDance Seed. 2025. Volcano engine reinforcement learning for llms.

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

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. 2022. Progprompt: Generating situated robot task plans using large language models. *Preprint*, arXiv:2209.11302.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *Preprint*, arXiv:2503.05592.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Prakhar Verma, Sukruta Prakash Midigeshi, Gaurav Sinha, Arno Solin, Nagarajan Natarajan, and Amit Sharma. 2025. Plan*rag: Efficient test-time planning for retrieval augmented generation. *Preprint*, arXiv:2410.20753.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *Preprint*, arXiv:2305.04091.

Ziting Wang, Haitao Yuan, Wei Dong, Gao Cong, and Feifei Li. 2024. Corag: A cost-constrained retrieval optimization system for retrieval-augmented generation. *Preprint*, arXiv:2411.00744.

Hui Wei, Zihao Zhang, Shenghua He, Tian Xia, Shijia Pan, and Fei Liu. 2025a. Plangenllms: A modern survey of llm planning capabilities. *Preprint*, arXiv:2502.11221.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, Hyokun Yun, and Lihong Li. 2025b. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. *Preprint*, arXiv:2505.16421.

xAI. 2025. Grok 4 deep research.

Renjun Xu and Jingwen Peng. 2025. A comprehensive survey of deep research: Systems, methodologies, and applications. *Preprint*, arXiv:2506.12594.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Tian Yu, Shaolei Zhang, and Yang Feng. 2024. Autorag: Autonomous retrieval-augmented generation for large language models. *Preprint*, arXiv:2411.19443.

Haoqi Yuan, Chi Zhang, Hongcheng Wang, Feiyang Xie, Penglin Cai, Hao Dong, and Zongqing Lu. 2023. Skill reinforcement learning and planning for open-world long-horizon tasks. *Preprint*, arXiv:2303.16563.

Zhenrui Yue, Honglei Zhuang, Aijun Bai, Kai Hui, Rolf Jagerman, Hansi Zeng, Zhen Qin, Dong Wang, Xuanhui Wang, and Michael Bendersky. 2025. Inference scaling for long-context retrieval augmented generation. *Preprint*, arXiv:2410.04343.

Dingchu Zhang, Yida Zhao, Jialong Wu, Baixuan Li, Wenbiao Yin, Liwen Zhang, Yong Jiang, Yufeng Li, Kewei Tu, Pengjun Xie, and Fei Huang. 2025. Evolvesearch: An iterative self-evolving search agent. *Preprint*, arXiv:2505.22501.

Huaixiu Steven Zheng, Swaroop Mishra, Hugh Zhang, Xinyun Chen, Minmin Chen, Azade Nova, Le Hou, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, and Denny Zhou. 2024a. Natural plan: Benchmarking llms on natural language planning. *Preprint*, arXiv:2406.04520.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. 2025. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. *Preprint*, arXiv:2504.03160.

Yuxiang Zheng, Shichao Sun, Lin Qiu, Dongyu Ru, Cheng Jiayang, Xuefeng Li, Jifan Lin, Binjie Wang, Yun Luo, Renjie Pan, Yang Xu, Qingkai Min, Zizhao Zhang, Yiwen Wang, Wenjie Li, and Pengfei Liu. 2024b. Openresearcher: Unleashing ai for accelerated scientific research. *Preprint*, arXiv:2408.06941.

A Implementation Details

In this work, we implement DEEPPLANNER using Qwen2.5-7B-Instruct[4] as the backbone model. For agentic reinforcement learning, we adopt asynchronous rollouts using `agent_loop` function[5] of `VERL` (Seed, 2025). Table 3 summarizes the hyperparameters used for training, and these values were maintained for all subsequent ablation experiments.

Table 3: Training configuration.

Parameter	Value
Training batch size (global)	64
Concurrent rollouts	8
Training steps	48
EAS coefficient α	0.1
Clipping factor κ	2
SAU coefficient β	2
Complexity threshold c	2
KL loss coefficient	0
Entropy coefficient	0
Clip ratio ϵ	0.2
Top-k	-1
Top-p	1
Temperature	1
Maximum context length	32,767

For DEEPPLANNER and its three variants (Vanilla, EAS, and SAU only), we train via full-parameter fine-tuning on a single node with 8 NVIDIA A100 (80GB) GPUs; each run completes in approximately 24 hours. At evaluation time, we set the decoding temperature to 0 to perform deterministic greedy decoding.

B Details of DEEPPLANNER

B.1 Tool Schema

Figure 12 presents the YAML Tool Schema Configuration employed by the DEEPPLANNER built upon the VERL architecture for external function calling. This configuration is essential for defining the interface and capabilities of the tools available to the agent.

B.2 System Prompt of DEEPPLANNER

As shown in Figure 9 and Figure 10, the prompt equips the agent with web search and browsing

[4] https://huggingface.co/Qwen/Qwen2.5-7B-Instruct

[5] https://verl.readthedocs.io/en/latest/advance/agent_loop.html

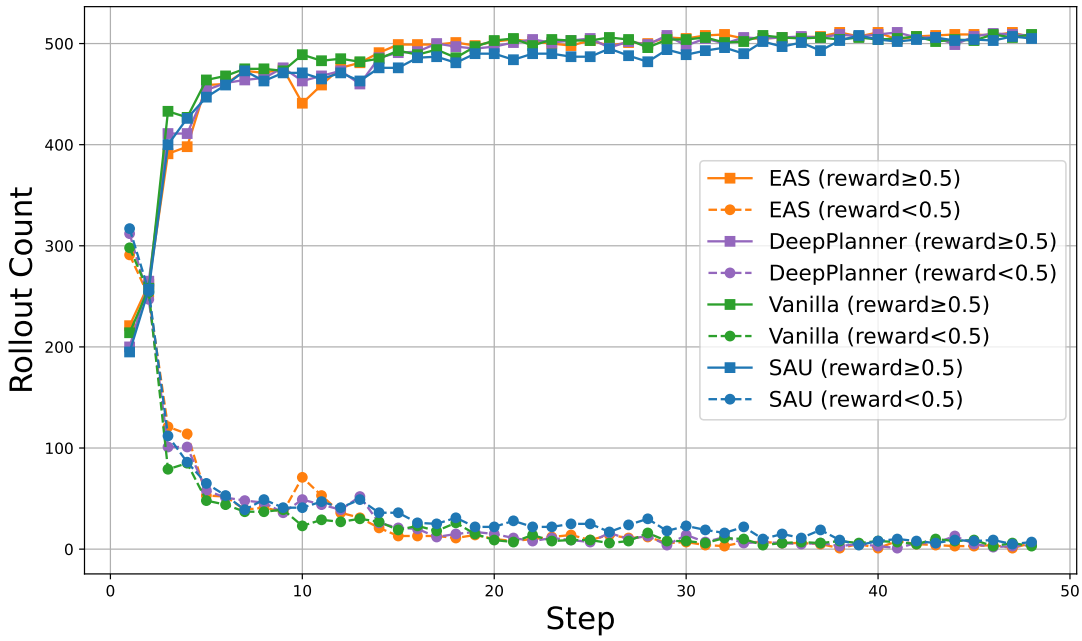

Figure 7: The average rollout reward over steps, where a reward less than 0.5 indicates a format error.

tools and mandates a structured workflow. Crucially, the agent must begin by outputting a structured `<plan></plan>`. Subsequent steps require strict adherence to defined output formats for tool calls (`<tool_call></tool_call>`) and final answers (`<answer></answer>`). In addition, the agent retains flexibility by being allowed to re-evaluate and update its plan based on new evidence gathered during the research process.

B.3 Prompt of Evaluation

We employ a high-fidelity evaluation prompt to assess the final answers of deep research agents, focusing on semantic correctness against established ground truths. As detailed in Figure 11, the prompt tasks a judge with comparing the agent's predicted answer to a list of acceptable ground truth answers. The core evaluation criterion is semantic equivalence, not direct lexical match. The evaluator is mandated to provide a structured JSON output containing a `"rationale"` and a binary `"judgement"` (`"correct"` or `"incorrect"`). This strict output directly feeds into our performance metrics: a `"correct"` judgment assigns a MBE score of 1, while an `"incorrect"` judgment assigns an MBE score of 0.

C Case Study

In detail, as shown in Figure 8, we remove the explicit planning stage from DeepPlanner (no longer requiring first-round (<plan> ... </plan>), keeping only (<think> ... </think>) and tool calls), retrain under same training set, rollout number, GRPO hyperparameters, and tool configurations, and select a representative failure caused by short-sighted planning. As shown in the left side of the figure, DeepPlanner first proposes a two-step plan: Step 1: identify the writer Nicholas O'Neill's father; Step 2: query the father's birthplace, then executes two targeted `web_search` calls guided by `sub_question` prompts and cross-checks Wikipedia and biography pages, yielding the correct answer: Pawtucket, Rhode Island, USA. In contrast, without an explicitly planning stage, the model only "plans" the first step and fails to consider global dependencies, mixing the end goal with missing intermediates: it starts from the writer's bio, triggers a brittle conditional chain, commits name concatenation errors (e.g., merging "Kane" with "O'Neill"), drifts to the unrelated entity "Evan O'Neill Kane," performs redundant and ineffective `web_search` calls, and outputs the wrong answer, Darby, Pennsylvania, USA. This contrast shows that explicit planning constrains the search space, stabilizes entity resolution, and clarifies verification. Without planning, short-sighted behavior accumulates linking and retrieval biases, precisely the failure mode our training addresses via entropy-shaped advantage on uncertain planning tokens and selective upweighting of high-quality, low-tool-call rollouts.

Question: What is the place of birth of Nicholas O'Neill (Writer)'s father?

DeepPlanner

<think>
To find the place of birth of Nicholas O'Neill's father, I need to first find out who his father is. Then, I will search for the place of birth of that person. Let's start by finding out who Nicholas O'Neill's father is.
</think>

Step-by-step plan

<plan>
First, search for information about the **father of Nicholas O'Neill**.
Then, search for the **place of birth of his father**.
</plan>

Step 1

<think>
To find out who Nicholas O'Neill's father is, I need to conduct a web search.
</think>

<tool_call>
{"name": "web_search", "arguments": {"query": ["**Who is Nicholas O'Neill's father**?"]}}
</tool_call>

<tool_response>
[{"search_query": "Who is Nicholas O'Neill's father?", "web_page_info_list": [
{"title": "Remembering The Station Nightclub Fire, More Than A Decade Later", "url": "https://thepublicsradio.org/article/remembering-station-nightclub-fire-more-decade-later/", "quick_summary": "**Nick O'Neill's father, Dave Kane**, said the longer it takes, the more likely for people will forget. "If you go down and visit the site, and ...", "browser": null, "sub_question": "Who is Nicholas O'Neill's father?"}
,... (other nine search results)]
</tool_response>

<think>
From the search results, we can see that Nicholas O'Neill's father is named Dave Kane. Now, I need to search for the place of birth of Dave Kane.
</think>

<tool_call>
{"name": "web_search", "arguments": {"query": ["**Place of birth of Dave Kane**"]}}
</tool_call>

Step 2

<tool_response>
[{"search_query": "Place of birth of Dave Kane", "web_page_info_list": [
{"title": "Dave Kane - Wikipedia", "url": "https://en.wikipedia.org/wiki/Dave_Kane", "quick_summary": "**Kane was born in Pawtucket, Rhode Island** and currently still makes his home in the Ocean State. Pawtucket, Rhode Island, U.S.", "browser": null, "sub_question": "Place of birth of Dave Kane"},
{"title": "Biography - Dave Kane", "url": "http://www.davekane.net/index.php?content=biography", "quick_summary": "Dave was born in Pawtucket, Rhode Island and currently still makes his home in the Ocean State.", "browser": null, "sub_question": "Place of birth of Dave Kane"}, ... other eight search results]
</tool_response>

<think>
Based on the search results, Dave Kane, the father of Nicholas O'Neill, was born in **Pawtucket, Rhode Island, USA**. Now I have the information to answer the question.
</think>

<answer>
Pawtucket, Rhode Island, USA
</answer>

without Plan

<think>
To find the place of birth of Nicholas O'Neill's father, I need to first identify who Nicholas O'Neill's father is. I will start by searching online for information about Nicholas O'Neill and identifying his father.
</think>

Short-sighted plan

<tool_call>
{"name": "web_search", "arguments": {"query": ["Nicholas O'Neill writer biography"]}}
</tool_call>
<tool_response>[...]</tool_response>

<think>
From the search result, the Wikipedia page and the Doollee.com page have the most relevant information. The Doollee.com page mentions that Nicholas O'Neill was the youngest of the 100 victims of the Station nightclub fire, which occurred in West Warwick, Rhode Island in February 2003. The Wikipedia page provides biographical information about Nicholas O'Neill, listing him as an American writer. However, it does not provide information about his father. I will search for the **place of birth of Nicholas O'Neill** to find more about his background.
</think>

Confusion of conditional

<tool_call>
{"name": "web_search", "arguments": {"query": ["Nicholas O'Neill (writer) place of birth"]}}
</tool_call>
<tool_response>
[{"search_query": "Nicholas O'Neill (writer) place of birth", "web_page_info_list": [
{"title": "Nicholas O'Neill Obituary (2003) - Providence, RI - Legacy.com", "url": "https://www.legacy.com/us/obituaries/providence/name/nicholas-o-neill-obituary?id=52156182", "quick_summary": "Nick was born in Warwick, a son of Joanne O'Neill of Pawtucket, formerly of Cranston, and **Kane**, of North Providence. He lived most of his ...", "browser": null, "sub_question": "Nicholas O'Neill (writer) place of birth"}]}]
</tool_response>

Incorrect name concatenation

<think>
I found that Nicholas O'Neill was born in Warwick, Rhode Island, USA. The obituary on Legacy.com has also provided information about his father, **Kane O'Neill**. I will continue to search to find information about Kane O'Neill's place of birth.
</think>
<tool_call>
{"name": "web_search", "arguments": {"query": ["Kane O'Neill place of birth"]}}
</tool_call>
<tool_response>
[{"search_query": "Kane O'Neill place of birth", "web_page_info_list": [
{"title": "Evan O'Neill Kane - Wikipedia", "url": "https://en.wikipedia.org/wiki/Evan_O%27Neill_Kane", "quick_summary": "Evan O'Neill Kane ; (1861-04-06)April 6, 1861 · Darby, Pennsylvania, US · April 1, 1932(1932-04-01) (aged 70). Kane, Pennsylvania, US · Jefferson Medical College.", "browser": null, "sub_question": "Kane O'Neill place of birth"}, {"title": "Kane O'Neill | Survivor Fanon Wiki - Fandom", "url": "https://survivorfanon.fandom.com/wiki/Kane_O%27Neill", "quick_summary": "Birth Date · Hometown · Occupation ; January 19, 1989 · Petersburg, VA · Personal Trainer ...", "browser": null, "sub_question": "Kane O'Neill place of birth"}]}]
</tool_response>

<think>
After searching further, I found that Kane O'Neill was born in Darby, Pennsylvania, which means he was born in the United States. However, we are looking for the place of birth of Nicholas O'Neill's father. I need to verify if Kane O'Neill is indeed the father of Nicholas O'Neill. The page from Doollee.com has mentioned that Kane O'Neill is the father of Nick O'Neill. Therefore, the place of birth of Nicholas O'Neill's father would be Darby, Pennsylvania, USA.
</think>
<answer>
Darby, Pennsylvania, USA
</answer>

Figure 8: In the case study, we remove the explicit planning stage from DEEPPLANNER and retrain the model under the same conditions. The results reveal that without planning, the model fails to consider global dependencies, leading to errors such as confusing forgotten conditions and drifting to unrelated entities. This highlights that explicit planning is crucial for effective entity resolution and verification, as it prevents short-sighted behavior that results in retrieval biases and incorrect answers.

DEEPPLANNER System Prompt

```
system_prompt = f"""
## Background Information
* Today is {current_date}
* You are Deep Research AI Assistant, an expert in conducting thorough, multi-step research.

The question I give you is a complex question that requires a deep research to answer.

To help you perform this task, you are equipped with two tools:
- A web search tool to help you perform google search.
- A webpage browsing tool to help you get new page content.

## Your Task
Do not answer the question immediately.
In the first step, you must output your plan inside <plan></plan> tags.
In later steps, you can use <tool_call></tool_call> to call tools or <answer></answer> to
 ↪  provide your final answer.
You can also re-evaluate and update your plan during the later steps.

## Output Format
You must strictly follow one and only one of the three output formats below at each step:

<think>
Your thinking process here.
</think>
<plan>
Step-by-step research plan or re-plan. Each step should be concise and action-oriented.
</plan>

or

<think>
Your thinking process here.
</think>
<tool_call>
Tool call with correct format.
</tool_call>

or

<think>
Your thinking process here.
</think>
<answer>
Final answer only — a word, phrase, or number.
If it's a yes-or-no question, respond with only "yes" or "no"
No explanations or additional commentary.
</answer>
"""
```

Figure 9: The system prompt used to instruct the agent for complex multi-step research tasks. The agent is required to first provide its plan in the initial turn, and subsequently execute it strictly, including tool calls and the final answer. The agent is also permitted to modify its plan based on the evidence gathered in previous steps.

DEEPPLANNER System Prompt (Continue)

```
# Tools

You may call one or more functions to assist with the user query.

You are provided with function signatures within <tools></tools> XML tags:
<tools>
{"type": "function", "function": {"name": "web_search", "description": "Search the web for
 ↪ relevant information from google. You should use this tool if the historical page content
 ↪ is not enough to answer the question. Or last search result is not relevant to the
 ↪ question.", "parameters": {"type": "object", "properties": {"query": {"type": "array",
 ↪ "description": "The queries to search"}}, "required": ["query"]}}}
{"type": "function", "function": {"name": "browse_webpage", "description": "Browse the webpage
 ↪ and return the content that not appeared in the conversation history. You should use this
 ↪ tool if the last action is search and the search result maybe relevant to the question.",
 ↪ "parameters": {"type": "object", "properties": {"url_list": {"type": "array",
 ↪ "description": "The chosen urls from the search result."}}, "required": ["url_list"]}}}
</tools>

For each function call, return a json object with function name and arguments within
 ↪ <tool_call></tool_call> XML tags:
<tool_call>
{"name": <function-name>, "arguments": <args-json-object>}
</tool_call>
```

Figure 10: The agent automatically appends the tool definitions (encased in `<tools>...</tools>`) to the base instructions of the system prompt.

Prompt for Final Answer Evaluation

```
You will be given a question and its ground truth answer list where each item can be a ground
 ↪ truth answer. Provided a pred_answer, you need to judge if the pred_answer correctly
 ↪ answers the question based on the ground truth answer list.
You should first give your rationale for the judgement, and then give your judgement result
 ↪ (i.e., correct or incorrect).

Here is the criteria for the judgement:
1. The pred_answer doesn't need to be exactly the same as any of the ground truth answers, but
 ↪ should be semantically same for the question.
2. Each item in the ground truth answer list can be viewed as a ground truth answer for the
 ↪ question, and the pred_answer should be semantically same to at least one of them.

question: {question}
ground truth answers: {gt_answer}
pred_answer: {pred_answer}

The output should in the following json format:

The output should in the following json format:
'''json
{
"rationale": "your rationale for the judgement, as a text",
"judgement": "your judgement result, can only be 'correct' or 'incorrect'"
}
'''
Your output:
```

Figure 11: The specialized prompt used to evaluate the final answer correctness of the deep research agent. When the final judgment is *correct*, the corresponding MBE score is assigned a value of 1; otherwise, the MBE score is 0.

DEEPPLANNER Tool Schema Yaml

```yaml
tools:
  - class_name: "user.tools.websearch_tool.WebSearchTool"
    config:
      type: native
    tool_schema:
      type: "function"
      function:
        name: "web_search"
        description: "Search the web for relevant information from google. You should use this
         ↪  tool if the historical page content is not enough to answer the question. Or last
         ↪  search result is not relevant to the question."
        parameters:
          type: "object"
          properties:
            query:
              type: "array"
              items:
                type: "string"
                description: "The query to search, which helps answer the question"
              description: "The queries to search"
          required: ["query"]
          minItems: 1
          uniqueItems: true

  - class_name: "user.tools.browse_tool.BrowseWebpageTool"
    config:
      type: native
    tool_schema:
      type: "function"
      function:
        name: "browse_webpage"
        description: "Browse the webpage and return the content that not appeared in the
         ↪  conversation history. You should use this tool if the last action is search and the
         ↪  search result maybe relevant to the question."
        parameters:
          type: "object"
          properties:
            url_list:
              type: "array"
              items:
                type: "string"
                description: "The chosen url from the search result, do not use url that not
                 ↪  appeared in the search result"
              description: "The chosen urls from the search result."
          required: ["url_list"]
```

Figure 12: The YAML tool schema configuration used by DEEPPLANNER under the VERL architecture.