WHEN EVERY ROLLOUT FAILS: SPLICING LEARNER-CONDITIONED OPTIONS INTO LONG-HORIZON AGENT RL

Anonymous authors
Paper under double-blind review

ABSTRACT

Group-based reinforcement learning with verifiable rewards often yields all-fail groups on long-horizon agent tasks: every sampled trajectory receives zero reward, leaving group-relative methods without a reward-based policy gradient. A common remedy supplies complete solutions from a stronger model, but full trajectories are expensive and expose the learner to long off-policy histories. We introduce SPLICE, which constructs *local* training targets from failed current-policy trajectories *online*. For each all-fail group, a proposal model selects a splice point in a current-policy trajectory and generates several bounded options from that point. We estimate each option's value using repeated current-policy continuations, convert positive estimates into a return-weighted local target, and fit the current policy on the options. We evaluate SPLICE across two policy families on two long-horizon agent tasks, Software Engineering with SWE-bench Verified and Deep Research with BrowseComp-Plus. SPLICE outperforms GRPO and full-trajectory guidance baselines in all four settings, improving over GRPO by 2.2–10.8 percentage points. On initially failed SWE tasks, it raises GRPO's graduation rate by 10.7 percentage points. Further experiments show that our method improves performance while reducing costly proposal-model computation.

1 INTRODUCTION

LLM agents solve long-horizon tasks through sequences of decisions and environment interactions (Yao et al., 2023; Zhou et al., 2024). Reinforcement learning with verifiable rewards (RLVR) trains these agents using outcome feedback, such as whether a code change passes a test suite. Group Relative Policy Optimization (GRPO) assigns advantages by comparing rewards within a group of trajectories sampled for the same task (Shao et al., 2024). With binary outcome rewards, a group in which every trajectory fails has zero reward variation and provides no reward-based policy gradient. We refer to this case as an *all-fail group*. Such a group leaves the policy without a successful trajectory to reinforce, even when some intermediate decisions could support a successful continuation.

External guidance offers a direct way to recover learning signals from all-fail groups by introducing successful behaviors that the current policy rarely generates. Existing methods for GRPO often derive such guidance from complete task-level solutions (Yan et al., 2025; Ma et al., 2026; Zhang et al., 2025). This choice treats a complete solution as the source of guidance. For long-horizon agents, however, a failure may be localized: a short corrective segment can lead to a state from which the current policy can complete the remaining task.

Thus, using complete solutions in this long-horizon setting incurs both computational and distributional costs. A complete long-horizon trajectory may span tens or hundreds of interaction turns (e.g., Qwen3.5-9B rollouts on our SWE tasks average 83.7 agent turns). Repeatedly generating such trajectories with a strong proposal model requires expensive inference. The proposal model is requested even for decisions that the current policy can already make. Also, using long pre-collected trajectories as supervision exposes the learner to off-policy risk, increasing the mismatch between the training distribution and the current policy's behavior (Luo et al., 2025; Rajaraman et al., 2020; Ross et al., 2011). These costs pose two design problems. The first is allocating proposal-model computation to bottlenecks encountered by current policy. The second is converting local proposals into a training target that reflects the current policy's ability to complete the task after each proposal.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We address both problems with SPLICE, which introduces failure-localized option splicing with current-policy continuation evaluation. For an all-fail group, a proposal model selects a splice point in one current-policy trajectory and generates several bounded options. We splice each option at this point, return control to the current policy, and sample repeated continuations to task completion. Their terminal outcomes estimate the option's current-policy continuation value, and we convert the positive values into a return-weighted local target. Only all-fail groups enter this splice branch, while other groups continue with standard GRPO. The bounded options save expensive proposal-model inference, while anchoring them in a current-policy history and supervising on the local target limits off-policy exposure.

We evaluate SPLICE on Software Engineering and Deep Research tasks. Across SWE-bench Verified and BrowseComp-Plus, it improves GRPO by 2.2–10.8 percentage points. On a fixed set of initially failed SWE tasks, it raises graduation rate from 7.7% to 18.4%. SPLICE uses fewer proposal-model rollout turns than full-trajectory guidance baselines while achieving better performance. Extensive experiments show that SPLICE can recover all-fail tasks without requiring the proposal model to generate or supervise complete trajectories.

Our contributions are as follows: **(1)** We introduce SPLICE, which formulates recovery from all-fail groups in long-horizon agent RL as learner-conditioned option splicing. This formulation allocates costly proposal-model computation only to splice points selected online. **(2)** We use current-policy continuation returns to define a target distribution over proposed options. By splicing bounded options into current-policy histories and fitting to this local distribution online, the update limits off-policy exposure relative to full-trajectory guidance. **(3)** We show across Software Engineering and Deep Research tasks that SPLICE improves performance and recovery of initially failed tasks while reducing proposal-model cost.

2 Related Work

2.1 Agentic RL and All-Fail Groups

Reinforcement learning has been extended from single-response reasoning to agents that act over multiple turns in stateful environments (Wang et al., 2025b; Yao et al., 2023). These settings often use sparse trajectory-level rewards from executable verifiers, which allow policy optimization over complete sequences of reasoning, actions, and environment observations. GRPO samples a group of trajectories for the same task and estimates each advantage from its return relative to the group, avoiding a learned value model (Shao et al., 2024). The resulting update depends on reward variation among trajectories sampled from the same task.

Under binary outcome rewards, an all-fail group has zero reward for every trajectory, so its reward-relative advantages vanish. All-fail groups contain no positive trajectory to reinforce, making exploration on hard tasks the central problem. Existing methods reallocate sampling or filter zero-variance groups (Yu et al., 2025; Zheng et al., 2025), construct advantages from failed responses (Le et al., 2026), or revise failures using gold answers (Guo et al., 2026). External guidance offers another route by proposing behaviors that the current policy rarely samples, which we review below.

2.2 External Guidance in RL

One line of work conditions RL rollouts on precomputed hints or prefixes, while optimizing only the policy-generated continuation (Zhang et al., 2025; Setlur et al., 2026; Qu et al., 2026; Ji et al., 2026; Yi et al., 2026; Nath et al., 2025). Another line trains directly on complete external solutions through supervised, off-policy, or mixed-policy objectives (Fu et al., 2026; Zhang et al., 2026; Ma et al., 2026; Yan et al., 2025). Such guidance is often derived from expert solutions, while direct supervision requires complete expert trajectories. Generating these trajectories is expensive for long-horizon agents, and expert-induced histories can differ from those visited by the current policy, creating covariate shift and compounding errors (Luo et al., 2025; Rajaraman et al., 2020). Concurrent work RSTG combines GRPO with on-policy distillation (OPD) on teacher solutions for all-fail groups (Han et al., 2026), requiring aligned teacher and student token spaces. SPLICE instead queries a proposal model online for short options at selected splice points in current-policy trajectories. Learner-conditioned continuation evaluation and local supervision reduce proposal-model cost and limit off-policy exposure.

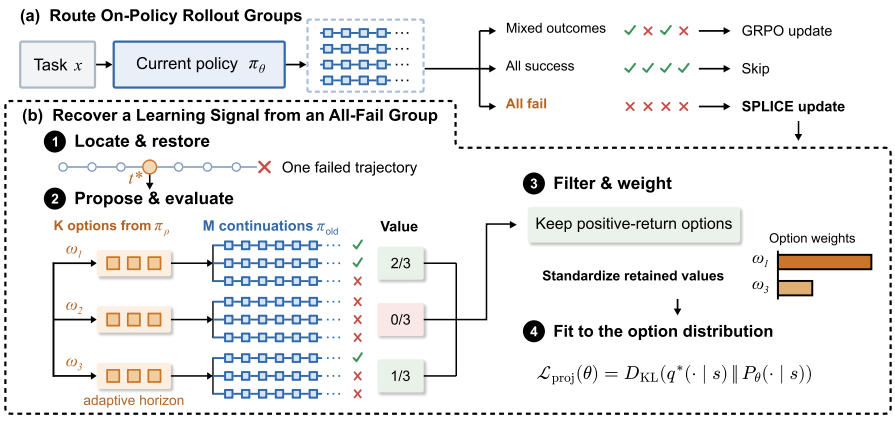

Figure 1: Method overview. (a) We route each group of G on-policy rollouts according to its binary outcomes. (b) The splice branch constructs a learner-conditioned target from short proposed options and current-policy continuations. We then fit the current policy to this target.

3 Preliminaries

Multi-turn agentic tasks. For a task x, let $s_t = (x, h_t, e_t)$ denote the interaction state at agent turn t, where h_t is the interaction history and e_t is the environment state. The policy samples an action $a_t \sim \pi_\theta(\cdot \mid s_t)$. A trajectory $\tau = (s_0, a_0, \ldots, a_{T-1}, s_T)$ receives a binary outcome reward $R(\tau) \in \{0, 1\}$.

GRPO training. At each update step, GRPO freezes a snapshot of the current policy π_θ, denoted by π_{old}, to collect trajectories while optimizing π_θ. For each task, it samples a group of G trajectories from π_{old} and estimates advantages without a learned value model (Shao et al., 2024). Let $\mathcal{G}(x) = \{\tau_i, R_i\}_{i=1}^G$ denote this group, where $R_i = R(\tau_i)$. The group-relative advantage is shared by all action tokens in trajectory τ_i:

$$\widehat{A}_i = \frac{R_i - \text{mean}\big(\{R_j\}_{j=1}^G\big)}{\text{std}\big(\{R_j\}_{j=1}^G\big) + \epsilon_A}. \tag{1}$$

Flatten the policy-generated tokens across all turns of τ_i as $y_{i,1:N_i}$, and let $c_{i,n}$ denote the full context available to the policy when generating $y_{i,n}$. With the token-level importance ratio $r_{i,n}(\theta) = \pi_\theta(y_{i,n} \mid c_{i,n}) / \pi_{\text{old}}(y_{i,n} \mid c_{i,n})$, the reward-dependent GRPO loss is

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\frac{1}{G} \sum_{i=1}^{G} \frac{1}{N_i} \sum_{n=1}^{N_i} \min\Big(r_{i,n}(\theta) \widehat{A}_i, \text{clip}(r_{i,n}(\theta), 1 - \epsilon_{\text{low}}, 1 + \epsilon_{\text{high}})\, \widehat{A}_i \Big). \tag{2}$$

This trajectory-level use of GRPO assigns the terminal outcome to action tokens across all agent turns (Wang et al., 2025b).

If every trajectory has the same binary reward, Equation 1 gives $\widehat{A}_i = 0$ for all i, so the reward-dependent gradient in Equation 2 vanishes. An all-success group already contains successful behavior, whereas an all-fail group provides no successful trajectory to reinforce. SPLICE targets the latter case.

4 Method

Overview. SPLICE supplements GRPO with a splice update for all-fail groups. Given an all-fail group, a proposal model π_ρ selects a saved state in one failed trajectory as the splice point and generates K bounded options from that state. We splice each option at this point and evaluate it with M continuations sampled from π_{old}. We then retain options with positive estimated return and fit π_θ to their return-weighted distribution. Figure 1 summarizes this process.

Update routing. We first route each rollout group $\mathcal{G}$ according to its observed outcomes:

$$\text{route}(\mathcal{G}) = \begin{cases} \text{GRPO}, & 0 < \sum_{i=1}^{G} R_i < G, \\ \text{SKIP}, & \sum_{i=1}^{G} R_i = G, \\ \text{SPLICE}, & \sum_{i=1}^{G} R_i = 0. \end{cases} \tag{3}$$

Mixed-reward groups use Equation 2, all-success groups are skipped, and all-fail groups enter the SPLICE branch described below.

4.1 Option Construction and Learner-Conditioned Evaluation

Splice-point selection. For each group $\mathcal{G}$ routed to SPLICE, we randomly select one failed trajectory τ^- from $\mathcal{G}$. The proposal model uses the task, full interaction history, and terminal feedback to select a saved state s_{t^*} as the splice point. It selects only states with saved snapshots. We then restore the corresponding environment state and interaction history. These snapshots are taken every C agent turns during rollout to save resources.

Option proposal. From s_{t^*}, the proposal model generates a candidate set of options $\mathcal{C}_{\pi_\rho}(s_{t^*}) = \{\omega_k\}_{k=1}^{K}$. Each option ω_k contains one or more agent turns generated by the proposal model. To bound the proposed segment, we limit each option to at most H_b consecutive turns. For the current GRPO rollout batch $\mathcal{B}_b$, we set H_b adaptively as

$$H_b = \max\left(1, \left\lceil \frac{\alpha}{|\mathcal{B}_b|} \sum_{\tau \in \mathcal{B}_b} T(\tau) \right\rceil\right), \tag{4}$$

where $T(\tau)$ is the number of agent turns in trajectory τ and $\alpha \in (0, 1]$ controls the option length relative to the mean rollout length.

Learner-conditioned evaluation. Evaluating the proposed options serves two purposes: filtering incorrect options and measuring whether the current policy can use each option to complete the task. Directly training on unverified options can reinforce proposal errors. Verifying them through proposal-model continuations addresses this issue, but they are costly and estimate option values under π_ρ. An option useful to π_ρ may still be ineffective for π_{old}. We therefore evaluate each non-terminal option through continuations from the current policy π_{old}. Their returns estimate learner-conditioned option values and define a local target distribution over the retained options. Current-policy weighting and supervision on bounded option segments together limit off-policy exposure.

Concretely, we execute each option from the same splice point s_{t^*}. If ω_k completes the task during proposal, we immediately apply the task verifier and set $\widehat{Q}_k = 1$ if it passes and $\widehat{Q}_k = 0$ otherwise. If the task remains incomplete after ω_k ends, π_{old} generates M independent continuations to task termination. Each resulting spliced trajectory $\tau_{k,j}$ consists of the prefix of τ^- before t^*, the interaction segment from ω_k, and the jth continuation. Its construction replaces the failed suffix with a bounded proposed option followed by a current-policy continuation. The mean reward across continuations estimates the learner-conditioned value of option ω_k:

$$\widehat{Q}_k = \frac{1}{M} \sum_{j=1}^{M} R(\tau_{k,j}). \tag{5}$$

We discard options with zero estimated return and skip the splice update if none remain. Let $\mathcal{I} = \{k : \widehat{Q}_k > 0\}$ denote the retained candidate set.

4.2 Return-Weighted Local Supervision

For the splice update, we normalize each retained option value against the retained set:

$$\widetilde{Q}_k = \frac{\widehat{Q}_k - \text{mean}\Big(\{\widehat{Q}_l\}_{l \in \mathcal{I}}\Big)}{\text{std}\Big(\{\widehat{Q}_l\}_{l \in \mathcal{I}}\Big) + \epsilon_Q}. \tag{6}$$

These normalized values serve as candidate-relative advantages and require no value model. We convert them into a return-weighted candidate distribution:

$$q^*(\omega_k \mid s_{t^*}) = \frac{\exp(\widetilde{Q}_k)}{\sum_{l\in\mathcal{I}} \exp(\widetilde{Q}_l)}, \tag{7}$$

where $k \in \mathcal{I}$.

Proposition 1 (Empirical dominance of return weighting)**.** *Fix a nonempty retained candidate set $\mathcal{I}$ and its empirical continuation returns $\{\widehat{Q}_k\}_{k\in\mathcal{I}}$. Let $u_k = 1/|\mathcal{I}|$ be the uniform target over this set. The return-weighted target in Equation 7 satisfies*

$$\sum_{k\in\mathcal{I}} q^*(\omega_k \mid s_{t^*})\widehat{Q}_k \geq \frac{1}{|\mathcal{I}|}\sum_{k\in\mathcal{I}} \widehat{Q}_k. \tag{8}$$

The inequality is strict if the retained empirical returns are not all equal.

Proposition 1 shows that, on the same retained candidates and observed continuation outcomes, return weighting assigns an empirical learner-conditioned value no lower than uniform supervision. This target-level result does not claim an improvement in the true return after a finite-sample policy update. Appendix A provides the proof.

Local policy update. We use a sequence-level KL projection to motivate fitting the local target. For brevity, we write $s = s_{t^*}$ and $q_k^* = q^*(\omega_k \mid s)$, with k indexing retained options. Let $P_\theta(\cdot \mid s)$ denote the option distribution induced by π_θ. The target q^* is supported only on the retained options. The sequence-level projection is

$$\mathcal{L}_{\text{proj}}(\theta) = D_{\text{KL}}(q^*(\cdot \mid s) \,\|\, P_\theta(\cdot \mid s)). \tag{9}$$

Holding q^* fixed and expanding each option likelihood over its action tokens gives

$$\mathcal{L}_{\text{proj}}(\theta) = -\sum_k q_k^* \sum_{n=1}^{N_k} \log \pi_\theta(y_{k,n} \mid c_{k,n}) + \text{const}. \tag{10}$$

Here, $y_{k,1:N_k}$ denotes the N_k action tokens within option ω_k, and $c_{k,n}$ contains all earlier tokens in the trajectory. The constant collects terms independent of θ. The sequence-level objective generally assigns a larger loss contribution to a longer option because it sums over its tokens. We therefore optimize the following length-normalized projection surrogate:

$$\mathcal{L}_{\text{splice}}(\theta) = \sum_k q_k^* \underbrace{\left[-\frac{1}{N_k}\sum_{n=1}^{N_k} \log \pi_\theta(y_{k,n} \mid c_{k,n})\right]}_{\text{mean token loss for option } \omega_k}. \tag{11}$$

This surrogate preserves the option-level target weights q_k^* while making each option contribute through its mean token loss. It covers only proposal-model action tokens within retained options. Unlike Equation 9, Equation 11 is not an exact KL projection over variable-length option sequences.

Training schedule. We keep GRPO and splice examples in separate mini-batches. For each GRPO rollout batch, we aggregate the eligible splice examples into a single splice mini-batch. We alternate training between GRPO and splice mini-batches. The proposal model and environment snapshots are used only during training.

5 Experiments

5.1 Experimental Setup

Tasks and data. For **Software Engineering (SWE)**, we combine subsets of SWE-Gym (Pan et al., 2025) and R2E-Gym (Jain et al., 2025), excluding tasks from repositories that overlap with SWE-bench. We evaluate resolve rate on SWE-bench Verified (Jimenez et al., 2024). For **Deep Research**, we use a dataset curated from open-source datasets with a quality-control pipeline, which Appendix B.2 describes. We evaluate answer accuracy on BrowseComp-Plus (Chen et al., 2026),

using Qwen3-Embedding-8B as the retriever and gpt-oss-20b as the judge model. To give all methods access to the same amount and distribution of training data, we sample a fixed subset of 3,200 tasks for both SWE and Deep Research. We report SWE and Deep Research performance in the main results, and further analyses use SWE.

Models. We study two policy families based on Qwen3-8B (Qwen Team, 2025) and Qwen3.5-9B (Qwen Team, 2026). Our Qwen3-8B experiments use a shared cold-start checkpoint to adapt to the scaffold interfaces and tool-call formats, denoted Qwen3-8B-CS, whereas Qwen3.5-9B starts directly from its released checkpoint. Cold-start details are in Appendix C. Thinking is disabled for Qwen3-8B-CS for better performance, and is enabled for Qwen3.5-9B. We use Qwen3.5-122B-A10B as the proposal model for both policy families. We report main results for both families, and further analyses use Qwen3-8B-CS as the current policy.

Agent scaffolds. For SWE, we use OpenHands (Wang et al., 2025a) as the scaffold. We keep the default system prompt and tools unchanged. For Deep Research, we customize the scaffold on Uni-Agent (Ding et al., 2026), providing three tools `search`, `open` and `find`. For all training methods, the turn limit is set to 150 and the context limit is set to 128k tokens.

Training configuration. Unless varied in ablations, we use $K = 3$ proposed options and $M = 3$ current-policy continuations per option. The adaptive option horizon uses $\alpha = 0.3$ in Equation 4. The snapshot interval is $C = 4$ agent turns. Appendix D reports the hardware allocation and other hyperparameters for training.

Comparison Setup. We consider SPLICE, SFT, RL, SFT+RL, and ReLIFT, starting from the same initial policy and using the common 3,200-task subset for each task domain. This protocol controls the available task data rather than method-specific computation, which we compare separately in Section 5.4. For **SFT**, the proposal model generates one full trajectory for every task in the common subset, and rejection sampling retains only successful trajectories for supervised fine-tuning. **RL** applies GRPO directly to the initial policy. **SFT+RL** initializes from the resulting SFT checkpoint and runs GRPO for the same number of RL steps as SPLICE. We adapt **ReLIFT** (Ma et al., 2026) to the online setting by requesting one full proposal-model trajectory for each task with an all-fail group and adding it to the buffer only upon verified success. Following the original method, ReLIFT triggers a fine-tuning step when the buffer contains at least 64 examples.

5.2 Main Results

Table 1: Main results on SWE-bench Verified and BrowseComp-Plus. Qwen3-8B-CS is the cold-start checkpoint described in Appendix C.

	Qwen3-8B-CS		*Qwen3.5-9B*	
Method	SWE-bench Verified	BrowseComp-Plus	SWE-bench Verified	BrowseComp-Plus
Initial policy	11.2	16.4	53.2	42.5
SFT	20.4	23.7	57.4	48.8
RL	16.2	19.0	58.4	47.2
SFT+RL	24.6	27.3	60.0	50.2
ReLIFT	20.2	22.8	59.2	48.4
SPLICE	**25.8**	**29.8**	**60.6**	**51.3**

Table 1 first shows the limitation of applying RL directly to the initial policy. RL improves performance in all four settings, but the gains over the initial policy are limited to 2.6–5.2 percentage points. This modest improvement is consistent with the sparse-reward exploration bottleneck: GRPO can reinforce sampled successes but obtains no reward-based gradient from all-fail groups.

Providing successful trajectories before RL yields substantially stronger results. SFT+RL outperforms direct RL by 1.6–8.4 points across the four settings, while the RL stage improves its SFT initialization by a further 1.4–4.2 points. These results show that RL achieves better performance after SFT has exposed the policy to successful task trajectories.

Online external guidance offers an alternative way to supply successful behavior during RL. ReLIFT improves over direct RL by 0.8–4.0 points, showing that online guidance improves final performance

in this comparison. SPLICE improves further over ReLIFT by 1.4–7.0 points and exceeds the higher-compute SFT+RL baseline by 0.6–2.5 points. It therefore achieves the highest performance in all four settings. These comparisons support the use of learner-conditioned local options rather than complete proposal trajectories for external guidance.

5.3 Recovering Failed Tasks

We assess whether proposal guidance helps recover failed tasks and examine how training gains vary with task difficulty.

Proposal Hit and Graduation Rates. The **proposal hit rate** is the fraction of proposal attempts on SWE training tasks that yield at least one verified successful trajectory. The **graduation rate** is the fraction of initially failed SWE training tasks solved by the trained policy. We evaluate each trained method with one rollout per task, and a task *graduates* if this rollout succeeds.

As shown in Table 2, SPLICE obtains a proposal hit rate of 31.7%, 22.4 percentage points below ReLIFT, while using 17.6% fewer proposal-model turns. Despite the lower hit rate, SPLICE graduates 18.4% of the initial-failure set, compared with 15.4% for ReLIFT and 7.7% for RL. The gains of 3.0 points over ReLIFT and 10.7 points over RL show that the splice updates yield stronger recovery in this experiment.

Table 2: Recovery and proposal-model cost on SWE training tasks. Proposal hit and graduation rates are percentages.

Method	Proposal hit	Graduation	π_ρ Turns
RL	N/A	7.7	0
ReLIFT	**54.1**	15.4	115.7k
SPLICE	31.7	**18.4**	95.3k

Recovery Across Task Difficulty. We compare training gains against standard GRPO across fixed Easy, Medium, and Hard subsets of the 3,200-task SWE training set; Appendix E defines the partition. Difficulty labeling and the evaluation in Table 3 use independent rollouts, so randomness in agent outcomes can yield initial-policy resolve rates below 100% on Easy tasks and above 0% on Medium and Hard tasks.

Table 3 shows that the largest improvement occurs on Medium tasks. Easy-task performance is already near saturation. On Medium tasks, SPLICE gains 23.0 points, exceeding RL's gain by 13.7 points. This gap suggests that local option splicing is particularly useful at intermediate difficulty, where standard RL makes limited progress but targeted corrections can still enable recovery. On Hard tasks, SPLICE gains 15.0 points compared with 6.2 points for RL.

Table 3: Resolve rates (%) by task difficulty on the SWE training set.

Method	Easy ($n = 342$)	Medium ($n = 1{,}087$)	Hard ($n = 1{,}771$)
Initial	99.4	0.5	0.2
RL	99.4	9.8	6.4
SPLICE	99.1	23.5	15.2

5.4 Training Dynamics and Cost

We analyze training along three complementary axes: resolve-rate dynamics over training updates, proposal-model rollout turns, and total wall-clock time. Proposal-model turns isolate generation by the expensive guidance model, whereas wall-clock time reflects the efficiency of the each method.

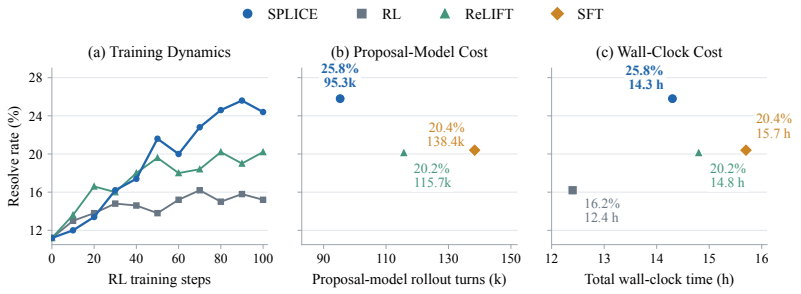

Figure 2: Training dynamics and cost. (a) Periodic resolve rate evaluated every 10 RL training steps. (b) Main-result resolve rate versus proposal-model rollout turns. (c) Main-result resolve rate versus total wall-clock time. Labels in (b,c) give resolve rate and cost, and higher and further left is better.

Training Dynamics. Figure 2(a) compares how resolve rate changes during training. RL improves quickly but soon stabilizes at a relatively low level. ReLIFT improves over a longer horizon, while SPLICE shows no early advantage but later surpasses both baselines. Thus, external guidance supports continued improvement, with SPLICE achieving the strongest late-stage performance.

Proposal-Model and Wall-Clock Cost. Figure 2(b,c) compares proposal-model generation and end-to-end training time. SFT collects a complete proposal-model trajectory for every task, giving it the highest proposal-model cost and longest runtime, whereas RL uses no proposal model and finishes fastest. ReLIFT and SPLICE spend additional time on online external guidance but outperform RL. For both methods, we asynchronously generate rollout batches for RL and supervised updates with *staleness*$= 2$ and observe comparable resolve rates to synchronous training; Appendix D provides configuration details. SPLICE requests only bounded proposal options rather than complete trajectories; despite the additional student continuations used to evaluate them, it requires less proposal-model computation and wall-clock time than ReLIFT and SFT while achieving the highest resolve rate.

5.5 Mechanism and Ablation Studies

We test four design choices: how continuation outcomes define the local target, which policy generates the continuations, how much of each proposal-generated branch enters the loss, and how long each option should be. All variants are trained under the same setting, except for the changes stated in each comparison. **Resolve** measures SWE-bench Verified performance; **Graduation** and **Hit** are training-set metrics defined in Section 5.3. Appendix F studies strategies for selecting splice points.

Continuation Evaluation and Target Construction. We compare SPLICE's return-weighted target with three alternatives. **Single option** directly supervises one generated option without further verification. **Hard top-$\widehat{Q}$** retains $K = M = 3$ continuation evaluation but places all target mass on the option attaining the largest positive $\widehat{Q}_k$. We randomly select one if multiple options have the same $\widehat{Q}_k$. **Uniform positive** uses the same continuation outcomes but assigns equal weight to every option with $\widehat{Q}_k > 0$. SPLICE improves Resolve by 3.8 points and Graduation by 6.1 points over single-option supervision. More directly, it exceeds hard top-$\widehat{Q}$ by 1.4 Resolve points and 1.4 Graduation points under matched proposal and continuation data. This result supports retaining a soft distribution over verified alternatives rather than collapsing the target onto the best. Relative to uniform positive, return-dependent weighting adds 2.2 Resolve points and 2.0 Graduation points, showing that the relative continuation returns provide useful information beyond binary outcome filtering.

Table 4: Mechanism ablations. Bold text marks the default configuration.

Continuation evaluation and option weighting				
Setting		Resolve	Graduation	
SPLICE		**25.8**	**18.4**	
Single option		22.0	12.3	
Hard top-$\widehat{Q}$		24.4	17.0	
Uniform positive		23.6	16.4	
Continuation policy and supervision scope				
Policy	Loss scope	Resolve	Graduation	π_ρ Turns
Current	**Local option**	**25.8**	**18.4**	**95.3k**
Proposal	Local option	26.2	18.7	374.2k
Proposal	Full suffix	22.8	14.8	358.9k
Adaptive option horizon				
α	Resolve	Hit	Graduation	π_ρ Turns
0.1	18.4	12.1	9.6	33.7k
0.3	**25.8**	**31.7**	**18.4**	**95.3k**
0.5	24.6	34.3	16.9	158.1k

Continuation Policy and Supervision Scope. We first replace current-policy continuations with **proposal-model continuations** while retaining supervision only on the bounded options. This change improves Resolve from 25.8% to 26.2% and Graduation from 18.4% to 18.7%, but raises proposal-model rollout turns from 95.3k to 374.2k, or $3.9\times$ as many turns. The **full-suffix variant** also supervises the action tokens in successful proposal-generated continuations. To control the contribution of sequence length, we average the token loss within each branch and then average successful branches within each option before applying its target weight. Full-suffix supervision lowers both Resolve and Graduation while using 358.9k proposal-model turns. The two variants therefore use comparable proposal-model computation, but extending supervision to the successful suffix degrades performance. This may be attributed to greater off-policy exposure along proposal-induced histories and accumulated covariate shift in long trajectories.

Adaptive Option Horizon. We vary the fraction in Equation 4 over $\alpha \in \{0.1, 0.3, 0.5\}$, with $\alpha = 0.3$ as the reference. **Reducing α to 0.1** cuts proposal-model turns from 95.3k to 33.7k, a 64.6% reduction, but lowers Hit, Resolve and Graduation significantly. This pattern suggests that

very short options are less likely to reach a state from which the current policy can complete the task. **Increasing α to 0.5** raises Hit by 2.6 points and proposal-model turns from 95.3k to 158.1k, a 65.9% increase. Resolve and Graduation nevertheless slightly decrease. Longer teacher options therefore do not yield a better proposal-free policy in this comparison.

5.6 Case Study: Redirecting an Incomplete Repair

We illustrate the splice update with SWE-Gym instance `dask__dask-10506` in Figure 3, which concerns a division-by-zero error when writing an empty array to Zarr. The raw steps are summarized for clarity.

An Incomplete Repair. We randomly select one trajectory in this all-fail group. It passes 471 existing regression tests and one of the two issue-specific tests, but fails `test_load_store_chunk`. In this trajectory, the agent reproduces the exception, inspects the shared write helper, and then changes Zarr chunk sizes from zero to one. This change fixes the reported example, but the helper still attempts to write empty blocks when called directly.

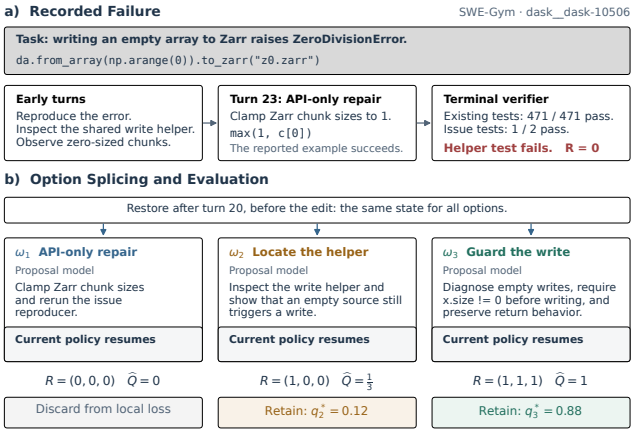

Figure 3: A case sampled from training with summarized trajectory steps. (a) Recorded actions and verifier feedback from a failed rollout. (b) Options from the proposal model, followed by current-policy continuations.

Option Splicing and Verification. The proposal model selects the saved state after 20 completed turns as the splice point, before the first production edit. All three options begin from this state. After each option, the current policy produces three independent continuations. The returns are $(0, 0, 0)$, $(1, 0, 0)$, and $(1, 1, 1)$ for ω_1, ω_2, and ω_3, respectively. Thus, the diagnostic option sometimes leads the current policy to the complete repair. These give $\widehat{Q} = (0, \frac{1}{3}, 1)$: the API-only option is discarded, while the diagnostic and guarded-write options remain. Equations 6–7 assign the retained options weights of approximately 0.12 and 0.88. The resulting target favors the guarded helper repair while retaining some probability on the diagnostic behavior that may also support a successful completion. This example illustrates how short, outcome-verified alternatives can extract a local training target from an all-fail group.

6 Limitations and Conclusion

Limitations. SPLICE requires environments that support restoring intermediate states, which may limit its application to live or irreversible interactions. It also introduces additional current-policy rollout computation because each option is evaluated through multiple continuations. Our asynchronous pipeline adds limited wall-clock time in our experiments, but the cost may be larger in environments with expensive execution. Finally, estimating each option value from three continuations introduces sampling error. Increasing its accuracy requires more current-policy continuations or a stronger model, creating a trade-off between estimation quality and inference cost.

Conclusion. We introduced SPLICE, a training method that recovers learning signals from all-fail rollout groups in long-horizon agent RL. It splices bounded proposal-model options into failed current-policy trajectories online. We evaluate each option through current-policy continuations and train on the resulting return-weighted local target. Replacing complete proposal trajectories with local options online reduces proposal-model cost and limits off-policy exposure and covariate shift. Across two policy families and two agent tasks, SPLICE outperforms GRPO and full-trajectory guidance, with particularly strong recovery on initially failed tasks.

AI USE DISCLOSURE

We used generative AI to assist with the construction of the Deep Research training dataset and with manuscript preparation. As detailed in Appendix B.2, DeepSeek-V4-Flash audited candidate query–answer pairs. During manuscript preparation, we used generative AI tools to revise wording, grammar, and presentation in parts of the paper. The authors reviewed all AI-assisted outputs, verified their factual and technical content, and revised them as needed. We take full responsibility for the content of this work.

REFERENCES

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Sahel Sharifymoghaddam, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Hosna Oyarhoseini, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. BrowseComp-Plus: A fair and disentangled evaluation benchmark for deep search agents. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22349–22370, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1023. URL https://aclanthology.org/2026.acl-long.1023/.

Zheng Chu, Xiao Wang, Jack Hong, Huiming Fan, Yuqi Huang, Yue Yang, Guohai Xu, Chenxiao Zhao, Cheng Xiang, Shengchao Hu, Dongdong Kuang, Ming Liu, Bing Qin, and Xing Yu. RED-Searcher: A scalable and cost-efficient framework for long-horizon search agents. *arXiv preprint arXiv:2602.14234*, 2026. URL https://arxiv.org/abs/2602.14234.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026. URL https://arxiv.org/abs/2606.19348.

Yuyang Ding, Bo Wen, Xubo Cao, Zhiqiang Zhai, Guangming Sheng, Xibin Wu, Juntao Li, Min Zhang, and Uni-Agent Contributors. Uni-Agent: Build, run, and train agents at scale. GitHub repository, 2026. URL https://github.com/verl-project/uni-agent.

Yuwen Du, Rui Ye, Shuo Tang, Xinyu Zhu, Yijun Lu, Yuzhu Cai, and Siheng Chen. OpenSeeker: Democratizing frontier search agents by fully open-sourcing training data. *arXiv preprint arXiv:2603.15594*, 2026. URL https://arxiv.org/abs/2603.15594.

Yuqian Fu, Tinghong Chen, Jiajun Chai, Xihuai Wang, Songjun Tu, Guojun Yin, Wei Lin, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. SRFT: A single-stage method with supervised and reinforcement fine-tuning for reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 102430–102459, 2026. URL https://openreview.net/forum?id=n6E0r6kQWQ.

Yiqiu Guo, Xueting Han, Qi Jia, Guangtao Zhai, and Jing Bai. LatentRevise: Learning from zero-hit reasoning. *arXiv preprint arXiv:2606.29938*, 2026.

Zhuowen Han, Jinwei Xiao, Zhengxi Lu, Renren Jin, Zhiyuan Yao, Yuxin Liu, Hongyan Hao, Yueqing Sun, Yu Yang, Qi Gu, Xunliang Cai, and Deyi Xiong. Distill where you fail: Recovering learning signals of negative RL-groups from adaptive teacher guidance. *arXiv preprint arXiv:2608.00782*, 2026.

Naman Jain, Jaskirat Singh, Manish Shetty, Tianjun Zhang, Liang Zheng, Koushik Sen, and Ion Stoica. R2E-Gym: Procedural environment generation and hybrid verifiers for scaling open-weights SWE agents. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=7evvwwdo3z.

Yuxiang Ji, Zengbin Wang, Yong Wang, Shidong Yang, Ziyu Ma, Guanhua Chen, Zonghua Sun, Liaoni Wu, and Xiangxiang Chu. Learning agentic policy from action guidance. *arXiv preprint arXiv:2605.12004*, 2026. URL https://arxiv.org/abs/2605.12004.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024.

Thanh-Long V. Le, Myeongho Jeon, Kim Vu, Viet Lai, and Eunho Yang. No prompt left behind: Exploiting zero-variance prompts in LLM reinforcement learning via entropy-guided advantage shaping. In *International Conference on Learning Representations*, volume 2026, pp. 121956–121982, 2026. URL https://openreview.net/forum?id=kiXFIESZKv.

Zhuofeng Li, Dongfu Jiang, Xueguang Ma, Haoxiang Zhang, Ping Nie, Yuyu Zhang, Kai Zou, Jianwen Xie, Yu Zhang, and Wenhu Chen. OpenResearcher: A fully open pipeline for long-horizon deep research trajectory synthesis. *arXiv preprint arXiv:2603.20278*, 2026. URL https://arxiv.org/abs/2603.20278.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 33:3776–3786, 2025. doi: 10.1109/TASLPRO.2025.3606231. URL https://doi.org/10.1109/TASLPRO.2025.3606231.

Lu Ma, Hao Liang, Meiyi Qiang, Lexiang Tang, Xiaochen Ma, Zhen Hao Wong, Junbo Niu, Chengyu Shen, Runming He, Yanhao Li, Bin Cui, and Wentao Zhang. Learning what reinforcement learning can't: Interleaved online fine-tuning for hardest questions. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 80802–80821, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/82c4ad0c3d3d4898a77be5df7b2ede3a-Paper-Conference.pdf.

Kyle Montgomery. Deep research SFT datasets. Hugging Face datasets, 2026. URL https://huggingface.co/kylemontgomery.

Vaskar Nath, Elaine Lau, Anisha Gunjal, Manasi Sharma, Nikhil Baharte, and Sean Hendryx. Adaptive guidance accelerates reinforcement learning of reasoning models. *arXiv preprint arXiv:2506.13923*, 2025. URL https://arxiv.org/abs/2506.13923.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with SWE-Gym. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. arXiv:2412.21139.

Yuxiao Qu, Amrith Setlur, Virginia Smith, Ruslan Salakhutdinov, and Aviral Kumar. POPE: Learning to reason on hard problems via privileged on-policy exploration. *arXiv preprint arXiv:2601.18779*, 2026. URL https://arxiv.org/abs/2601.18779.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents. https://qwen.ai/blog?id=qwen3.5, 2026.

Nived Rajaraman, Lin Yang, Jiantao Jiao, and Kannan Ramchandran. Toward the fundamental limits of imitation learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 2914–2924, 2020. URL https://arxiv.org/abs/2009.05990.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 627–635, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

Amrith Setlur, Zijian Wang, Andrew Cohen, Paria Rashidinejad, and Sang Michael Xie. Reuse your FLOPs: Scaling RL on hard problems by conditioning on very off-policy prefixes. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=a5itZI3DeQ. arXiv:2601.18795.

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

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Daniel Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=OJd3ayDDoF.

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

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 117157–117186. Curran Associates, Inc., 2025. doi: 10.52202/085713-3904. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/a9d5c33e38442450f9781bd7bad3c81e-Paper-Conference.pdf.

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

Junkeun Yi, Damon Mosk-Aoyama, Baihe Huang, Ritu Gala, Charles Wang, Sugam Dipak Devare, Khushi Bhardwaj, Abhibha Gupta, Oleksii Kuchaiev, Jiantao Jiao, Jian Zhang, and Venkat Srinivasan. PivotRL: High accuracy agentic post-training at low compute cost. *arXiv preprint arXiv:2603.21383*, 2026. URL https://arxiv.org/abs/2603.21383.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 113222–113244. Curran Associates, Inc., 2025. doi: 10.52202/085713-3775. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/a4277440d50f1f15d2cb4c14f7e0c0d2-Paper-Conference.pdf.

Wenhao Zhang, Yuexiang Xie, Yuchang Sun, Yanxi Chen, Guoyin Wang, Yaliang Li, Bolin Ding, and Jingren Zhou. On-policy RL meets off-policy experts: Harmonizing supervised fine-tuning and reinforcement learning via dynamic weighting. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 120693–120726, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/c44a04289beaf0a7d968a94066a1d696-Paper-Conference.pdf.

Xuechen Zhang, Zijian Huang, Yingcong Li, Chenshun Ni, Jiasi Chen, and Samet Oymak. BREAD: Branched rollouts from expert anchors bridge SFT & RL for reasoning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 96726–96752. Curran Associates, Inc., 2025. doi: 10.52202/085713-3235. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/8b69f0fac9b83f5ba43c7a851a841a61-Paper-Conference.pdf.

Haizhong Zheng, Yang Zhou, Brian R. Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act only when it pays: Efficient reinforcement learning for LLM reasoning via selective rollouts. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 124321–124346. Curran Associates, Inc., 2025. doi: 10.52202/

085713-4143. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/b42fb82fc3ffb88b872c4714f093875d-Paper-Conference.pdf.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

Yuhang Zhou, Kai Zheng, Qiguang Chen, Mengkang Hu, Qingfeng Sun, Can Xu, and Jingjing Chen. OffSeeker: Online reinforcement learning is not all you need for deep research agents. *arXiv preprint arXiv:2601.18467*, 2026. URL https://arxiv.org/abs/2601.18467.

APPENDIX

A PROOF OF PROPOSITION 1

Proof. Let $m = |\mathcal{I}|$ and $u_k = 1/m$. Define $d = \mathrm{std}(\{\widehat{Q}_l\}_{l\in\mathcal{I}}) + \epsilon_Q > 0$ and $\overline{Q} = m^{-1}\sum_{k\in\mathcal{I}} \widehat{Q}_k$. The normalized return then satisfies $\widetilde{Q}_k = (\widehat{Q}_k - \overline{Q})/d$. Consider the KL-regularized objective

$$F(q) = \sum_{k\in\mathcal{I}} q_k \widetilde{Q}_k - D_{\mathrm{KL}}(q\|u), \qquad q \in \Delta(\mathcal{I}). \tag{12}$$

This objective is strictly concave in q. Its Lagrangian stationarity condition gives

$$q_k \propto u_k \exp(\widetilde{Q}_k). \tag{13}$$

Because u is uniform, the unique maximizer is exactly the softmax target q^* in Equation 7. Since u is feasible, optimality gives

$$F(q^*) \geq F(u), \tag{14}$$

$$\sum_{k\in\mathcal{I}} q_k^* \widetilde{Q}_k \geq \sum_{k\in\mathcal{I}} u_k \widetilde{Q}_k + D_{\mathrm{KL}}(q^*\|u). \tag{15}$$

The affine relation $\widehat{Q}_k = \overline{Q} + d\widetilde{Q}_k$ preserves comparisons between expectations under distributions on the fixed set $\mathcal{I}$. Dropping the nonnegative KL term in Equation 15 therefore yields Equation 8.

If the retained empirical returns are not all equal, then $\widetilde{Q}$ is nonconstant and its softmax is nonuniform. Hence $q^* \neq u$ and $D_{\mathrm{KL}}(q^*\|u) > 0$. In this case, Equations 15 and 8 are strict. If all retained empirical returns are equal, then $q^* = u$ and equality holds. □

B TRAINING DATA

B.1 SOFTWARE ENGINEERING

We construct the SWE training pool from subsets of SWE-Gym (Pan et al., 2025) and R2E-Gym (Jain et al., 2025). SWE-Gym collects real-world issue-resolution tasks from pull requests in Python repositories and pairs each task with an executable environment and expert-validated tests (Pan et al., 2025). It contains 2,438 tasks, while we omit 31 tasks because their container images are unavailable. R2E-Gym procedurally constructs executable tasks from repository commits through back-translation and test collection or generation (Jain et al., 2025). We use its subset of 4,578 tasks from repositories that do not overlap with SWE-bench.

B.2 DEEP RESEARCH DATASET CONSTRUCTION

We draw candidate tasks from five public releases, which are OpenResearcher (Li et al., 2026), DeepForge from OffSeeker (Zhou et al., 2026), OpenSeeker (Du et al., 2026), REDSearcher (Chu et al., 2026) and public Deep Research SFT datasets released by Kyle Montgomery (Montgomery, 2026). Some releases provide complete interaction traces in addition to question–answer pairs. We extract only the query and reference answer.

We canonicalize each query with Unicode NFKC normalization, case folding, and whitespace collapsing. We then deduplicate the normalized queries jointly across all five sources, obtaining 75,372 unique candidates.

Collection dates and retrieval systems differ across these releases, so a reference answer may be ambiguous, outdated, or unsupported in our web environment. We audit every candidate with DeepSeek-V4-Flash (DeepSeek-AI, 2026) under a unified environment. The model can call `search`, `open`, and `find`, and Serper provides web search and reading services.[1] We accept a task only when the audit answer agrees with its reference and the current search results provide supporting evidence. This common audit removes ill-posed questions, stale answers, and tasks whose evidence is no longer reachable. The final pool contains 16,855 Deep Research training tasks.

[1] `https://serper.dev/`

C COLD-START TRAINING FOR QWEN3-8B

We apply cold start only to Qwen3-8B. This stage adapts the released checkpoint to the agent scaffolds and their tool-call formats. Qwen3.5-9B starts directly from its released checkpoint.

We sample 500 SWE tasks and 500 Deep Research tasks from the data pool described in Appendix B. Qwen3.5-122B-A10B generates candidate trajectories, and rejection sampling retains trajectories that pass the corresponding verifier. We mix the retained SWE and Deep Research trajectories into one set and fine-tune the released Qwen3-8B checkpoint to obtain Qwen3-8B-CS.

Table 5 separates the effect of cold start from subsequent training. Under the evaluation setup used in the main experiments, the released Qwen3-8B checkpoint obtains 8.6% on SWE-bench Verified and 12.7% on BrowseComp-Plus. Qwen3-8B-CS obtains 11.2% and 16.4%, respectively. All Qwen3-8B methods in Table 1 start from this same Qwen3-8B-CS checkpoint.

Table 5: Qwen3-8B performance before method-specific training.

Initial checkpoint	SWE-bench Verified (%)	BrowseComp-Plus (%)
Qwen3-8B (released)	8.6	12.7
Qwen3-8B-CS	11.2	16.4

D TRAINING CONFIGURATION

We train on four compute nodes, each containing 8 NVIDIA H200 GPUs. For the RL baseline, all four nodes run the student policy, with the trainer and rollout actor colocated on the same GPUs. For SPLICE and ReLIFT, three nodes use this colocated student setup, while the remaining node serves the proposal model. Training is asynchronous with a maximum rollout staleness of 2. Each rollout batch contains 32 groups, with $G = 8$ trajectories per group, and the actor learning rate is 2×10^{-6}. For the SFT baseline, we train for three epochs with a learning rate of 1×10^{-5}.

E TASK-DIFFICULTY PARTITION

We construct fixed difficulty subsets from the 3,200-task SWE training set before training. We run Qwen3-8B-CS and Qwen3-Coder-30B-A3B-Instruct once on each task and assign labels from their verified outcomes. A task is *Easy* if Qwen3-8B-CS succeeds, regardless of the Qwen3-Coder-30B-A3B-Instruct outcome. Among the remaining tasks, those solved by Qwen3-Coder-30B-A3B-Instruct are *Medium*, and those on which both models fail are *Hard*. The labeling runs yield success rates of 10.7% for Qwen3-8B-CS and 43.6% for Qwen3-Coder-30B-A3B-Instruct, with 308 tasks solved by both models. The resulting partition contains 342 Easy, 1,087 Medium, and 1,771 Hard tasks.

F SPLICE-POINT SELECTION

We compare three rules for choosing the splice point using the setup in Section 5.5. Each rule samples one failed trajectory uniformly from the all-fail group and restricts selection to states with saved snapshots, with $C = 4$. *Random* selects an eligible splice point uniformly. *Highest entropy* selects the eligible splice point whose recorded action has the largest mean token entropy under π_{old}. For the N_t action tokens at turn t, we compute

$$\mathcal{H}_t = -\frac{1}{N_t} \sum_{n=1}^{N_t} \sum_{v \in \mathcal{V}} \pi_{\text{old}}(v \mid c_{t,n}) \log \pi_{\text{old}}(v \mid c_{t,n}), \tag{16}$$

where $\mathcal{V}$ is the policy vocabulary and $c_{t,n}$ is the context before token n in that turn. Averaging over tokens avoids ranking turns by action length; ties select the earliest eligible turn. *Proposal model* uses the proposal model's splice-point selector from Section 4.1.

Table 6: Splice-point selection ablation. Resolve is the SWE-bench Verified resolve rate, and Graduation is the success rate on initially failed SWE training tasks. Both are percentages. Each selector operates only at saved snapshot boundaries.

Splice-point selection	Resolve	Graduation
Random	24.6	17.0
Highest entropy	24.0	16.2
Proposal model	**25.8**	**18.4**

Highest-entropy selection lowers Resolve by 0.6 points and Graduation by 0.8 points relative to random selection (Table 6), indicating that local policy uncertainty alone does not identify better splice points. Proposal-model selection improves Resolve by 1.2 points and Graduation by 1.4 points over random selection; relative to highest-entropy selection, the gains are 1.8 and 2.2 points, respectively. These results suggest that conditioning on the full failed trajectory and terminal feedback provides a more useful selection signal than either random choice or local token entropy.