VEL: VERIFIABLE EXPERIENTIAL LEARNING FOR RECURSIVE SELF-IMPROVING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) agents are increasingly applied to long-horizon tasks that require sustained interactions and iterative decision-making. In such environments, agents often need to learn from their previous attempts, improving future behaviors by extracting and reusing useful knowledge from past experiences. This capability, known as experiential learning, is essential for enabling agents to progressively adapt and improve beyond their initial capabilities. Recent approaches have sought to enhance experiential learning through reinforcement learning (RL), yet they typically rely on coarse-grained and implicit outcome-based rewards, providing limited guidance on how agents should leverage prior experiences. In this paper, we propose VEL, an RL framework for verifiable experiential learning. VEL isolates the experiential learning capability through a clear cross-episode formulation: the agent makes multiple sequential attempts on the same task and leverages experience distilled from previous attempts to guide future decisions. To reinforce experiential learning with fine-grained and verifiable feedback, VEL introduces the experience-guided advancement reward (EGAR), which is derived from improvements in verifiable task progress between consecutive episodes. Experiments on four representative benchmarks show that VEL outperforms various RL baselines and existing experiential learning approaches, yielding an average relative improvement of 49.3% from the first to the third attempt by leveraging prior episodes. We further demonstrate that agents trained with VEL in a single environment exhibit transferable experiential learning capabilities in out-of-distribution environments, enabling both same-task generalization and cross-task transfer. These results indicate that VEL equips LLM agents with transferable experiential learning capabilities, supporting recursive self-improving agents.

1 INTRODUCTION

Large language model (LLM) agents are increasingly applied to long-horizon tasks that require sustained interaction with complex environments (OpenAI, 2026; Anthropic, 2026), such as open-world exploration (Wang et al., 2024; Zhu et al., 2023), software engineering (Jimenez et al., 2024; Ding et al., 2025), and computer control (Xie et al., 2024; Yuan et al., 2026). During interaction, agents inevitably encounter failures, unexpected outcomes, and hidden constraints, making it necessary to learn from environmental feedback (Yao et al., 2023; Shinn et al., 2023). This motivates experiential learning: the ability to distill reusable knowledge from past interactions to improve future decisions (Zhao et al., 2024; Ouyang et al., 2025).

Recent studies have explored experiential learning as a trainable capability and sought to enhance it through reinforcement learning (RL) (Wang et al., 2026a; Shi et al., 2026a; Zhang et al., 2026b; Xia et al., 2026). A common paradigm implements experiential learning through a multi-stage pipeline: experiences are distilled from historical trajectories, stored in an external bank (Wang et al., 2025b), and retrieved by an auxiliary retrieval module to augment future task-execution (Shi et al., 2026b; Zhang et al., 2026a). Some methods optimize experience distillation indirectly using the binary success of the resulting experience-augmented execution as a proxy for the effectiveness of the distilled experience (Muhtar et al., 2026). However, we argue that this paradigm has two fundamental limitations. First, the outcome of an experience-augmented attempt cannot reliably reflect the effectiveness of the distilled experience, as it is jointly determined by multiple factors,

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

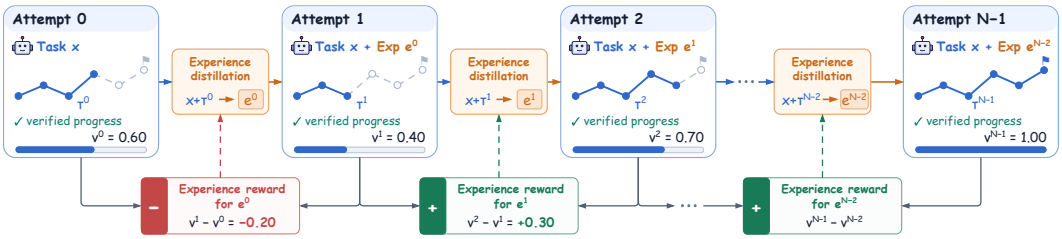

Figure 1: Overview of VEL. The agent makes successive attempts on the same task and, after each attempt, distills a textual experience from its trajectory to guide the next attempt. Each experience is rewarded by the change in verified task progress between the two attempts, which is positive if the experience helps the next attempt advance further (green) and negative otherwise (red).

including the base model's task-execution capability, task difficulty, and intermediate experience-management components such as storage and retrieval. Second, binary task-success rewards are inherently coarse-grained (Lightman et al., 2024; Feng et al., 2025; Wang et al., 2026b): they fail to capture partial improvements enabled by distilled experience, which may substantially advance task progress without immediately leading to final success. Consequently, the contribution of the distilled experience can be neither reliably attributed nor precisely quantified, leading to ambiguous credit assignment and hindering reliable optimization of the experience-distillation process.

To address these challenges, we propose VEL, a cross-episode RL framework for **V**erifiable **E**xperiential **L**earning. VEL formulates experiential learning as a sequential cross-episode process, where an agent makes multiple successive attempts on the same task and leverages experience distilled from previous episodes to guide subsequent attempts (Figure 1). This formulation provides a controlled setting for experiential learning, reducing the confounding introduced by multi-stage experience-processing pipelines. Rather than evaluating an experience solely by whether the subsequent attempt ultimately succeeds, we assess whether conditioning on that experience enables the agent to achieve additional progress beyond the preceding attempt. Such relative advancement captures useful intermediate improvements overlooked by binary task-success rewards, providing a finer-grained and more directly attributable learning signal. Accordingly, we optimize experiential learning with an objective defined by the relative advancement between consecutive attempts.

Specifically, VEL implements this objective through a shared policy that jointly performs task-execution and experience distillation. In each episode, the agent interacts with the environment and extracts reusable experience from its trajectory to guide subsequent attempts. This process repeats until the task is solved or the maximum number of attempts is reached. To quantify the improvement achieved between attempts, VEL defines a verifiable advancement signal (VAS) based on fine-grained environment states. The difference between VAS values of two consecutive episodes defines the Experience-Guided Advancement Reward (EGAR), which provides direct supervision for optimizing experiential learning. By jointly optimizing task-execution and experience distillation, VEL encourages the agent to extract useful experience and improve future decisions.

We evaluate VEL on four representative benchmarks: MineSweeper (Jiang et al., 2025), Sokoban (Racanière et al., 2017), WebShop (Yao et al., 2022), and ScienceWorld (Wang et al., 2022). VEL consistently outperforms prompting-based methods, standard RL baselines, and existing experiential learning approaches, achieving the strongest multi-attempt performance across all four benchmarks. In particular, VEL obtains the highest $p@2$ and $p@3$ in every environment and achieves the largest absolute gain from the first to the third attempt, corresponding to an average relative improvement of 49.3%. These results demonstrate that VEL enables agents to benefit substantially from the experience distilled from their prior attempts, rather than merely improving single-attempt task-execution. We further show that the experiential learning capability acquired by a model trained with VEL solely on MineSweeper transfers directly to the previously unseen ALFWorld environment (Shridhar et al., 2021), without additional training or fine-tuning. Despite being trained with only three attempts, the model continues to improve steadily when evaluated over five attempts. More notably, the experiential learning capability acquired from repeated attempts on the same task also transfers to sequential learning across related but distinct tasks. Together, these findings support treating experiential learning as a transferable atomic capability and point toward long-horizon agents that can iteratively self-improve through experience.

2 Related Works

Reinforcement Learning for LLM Agents. Reinforcement learning with verifiable rewards (RLVR) has emerged as a central paradigm for training LLM agents (Lambert et al., 2024; Shao et al., 2024; Wang et al., 2025a). To equip LLM agents for more complex, long-horizon tasks, recent works can be broadly divided into two directions. The first directly optimizes the task-solving policy. GiGPO introduces anchor-state-based two-level advantage estimation for fine-grained credit assignment (Feng et al., 2025), HGPO constructs context-aware hierarchical groups for more reliable step-level credit estimation (He et al., 2026), and BEACON further adopts milestone-guided trajectory segmentation to mitigate long-horizon reward misattribution (Wang et al., 2026b). Another line of work decomposes task-solving into distinct agentic capabilities, including planning (Tang et al., 2025; Lu et al., 2025), search (Jin et al., 2025; Song et al., 2025), exploration (Jiang et al., 2025; Ye et al., 2026) and self-verification (Chen et al., 2026a), and optimize them independently. VEL introduces a cross-episode RL formulation that explicitly decouples the experiential learning capability, enabling it to be accurately evaluated and effectively optimized.

Experiential Learning for LLM Agents. To enable continuous adaptation in complex long-horizon tasks (Jimenez et al., 2024; Chen et al., 2026b; Yang et al., 2026), LLM agents are expected to learn from past interactions. Early methods typically rely on structured prompting pipelines (Yao et al., 2023; Shinn et al., 2023), where agents are required to explicitly reflect after each step or attempt before making subsequent decisions. With the development of RLVR, a growing body of work has begun to formulate experiential learning as a trainable capability. SAGE (Wang et al., 2026a) and ERL (Shi et al., 2026a) conduct two sequential rollouts over the same or related tasks: experience distilled from the first rollout is used to guide the next rollout. The final outcome then serves as supervision for reinforcing experience extraction. To enable experience reuse across tasks, several methods construct end-to-end pipelines that encompass experience extraction, storage, management, and retrieval (Shi et al., 2026b; Zhang et al., 2026b;a; Muhtar et al., 2026). These methods optimize experience extraction using signals derived from downstream task outcomes (Muhtar et al., 2026), experience utilization (Zhang et al., 2026b), or self-assessment (Zhang et al., 2026a), thereby allowing the experience extractor and actor policy to co-evolve (Muhtar et al., 2026). Although these methods reinforce the capacity for experiential learning, they heavily rely on noisy or implicit signals. In contrast, VEL establishes a rigorous cross-episode RL formulation together with EGAR, providing direct, fine-grained, and attributable learning objectives for experiential learning.

3 Preliminaries

3.1 Task Formulation

We formulate the interaction between an LLM agent and a task environment as a partially observable Markov decision process (POMDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{Z})$, where $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{O}$ denote the state, action, and observation spaces, respectively. At each step, the agent cannot access the underlying state $s_t \in \mathcal{S}$ and receives only a partial observation $o_t \sim \mathcal{Z}(\cdot \mid s_t)$. Given a task instruction x sampled from a task distribution $\mathcal{D}$, the agent selects an action $a_t \sim \pi_\theta(\cdot \mid h_t, x)$ conditioned on the interaction history $h_t = (o_0, a_0, \ldots, o_t)$. The action induces a transition $s_{t+1} \sim \mathcal{P}(\cdot \mid s_t, a_t)$, producing a new observation. This process generates a trajectory $\tau = (o_0, a_0, \ldots, o_T)$, which terminates upon task completion or the step budget is exhausted. In standard RLVR, each trajectory receives a binary task-success reward $R(\tau) \in \{0, 1\}$. The standard learning objective is to maximize the expected task success $\mathcal{J}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, \tau\sim\pi_\theta(\cdot|x)}[R(\tau)]$.

3.2 Experiential Learning

In experiential learning, the agent reflects on its history $\mathcal{H}$ and distills a textual experience $e \sim \pi_\theta(\cdot \mid x, \mathcal{H})$, which then guides task-execution to produce a new trajectory $\tau \sim \pi_\theta(\cdot \mid x, e)$. Thus, experiential learning comprises two coupled operations: extracting information useful for future decisions and executing the task under the extracted experience. The objective is

$$\mathcal{J}_{\text{exp}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, e\sim\pi_\theta(\cdot|x,\mathcal{H}),\, \tau\sim\pi_\theta(\cdot|x,e)}\left[R(\tau)\right]. \tag{1}$$

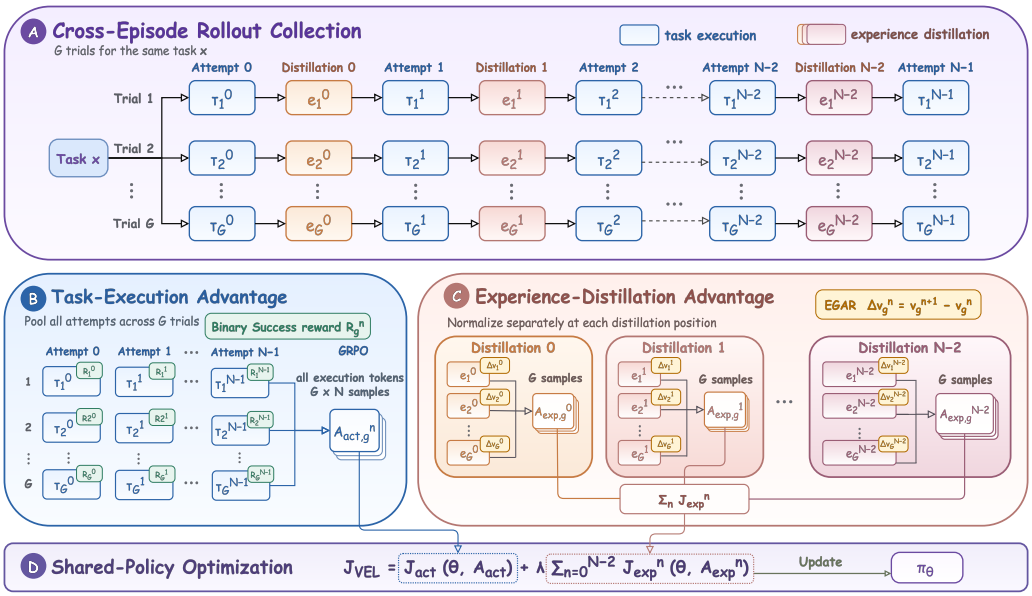

Figure 2: Overview of the VEL training framework. (a) Collection of a batch of G cross-episode trials for the same task. (b) Task-execution advantage estimation jointly across all attempts using binary task-success rewards. (c) Transition-wise advantage estimation for experience distillation using EGAR. (d) Joint optimization of both objectives with a shared policy.

Most existing work assigns the binary task-success reward $R(\tau)$ to both task-execution and experience-distillation tokens, and then uses standard RL to optimize the two processes jointly:

$$\nabla_\theta \mathcal{J}_{\text{exp}}(\theta) = \mathbb{E}\left[R(\tau)\nabla_\theta \log \pi_\theta(\tau \mid x, e) + R(\tau)\nabla_\theta \log \pi_\theta(e \mid x, \mathcal{H})\right]. \tag{2}$$

However, the binary task-success reward $R(\tau)$ is an inadequate learning signal for experience distillation. First, the outcome of the experience-guided trajectory is jointly influenced by the quality of the distilled experience e, the task-execution capability of π_θ, the difficulty of task x, and other factors. Second, $R(\tau)$ takes a binary value that indicates whether the trajectory reaches final success, and therefore does not represent partial improvements enabled by e before task completion.

4 Method

We present VEL, an RL framework for verifiable experiential learning. VEL addresses two central challenges. (1) The contribution of distilled experience is often entangled with the base model's task-execution capability, task difficulty, and auxiliary memory-management components, making it difficult to evaluate and optimize in isolation. To address this challenge, we introduce a cross-episode formulation to isolate experiential learning, as described in Section 4.1. (2) Binary task-success rewards cannot capture experiences that advance task-execution but are insufficient for final success. To provide a clear and fine-grained learning signal, we propose EGAR, as introduced in Section 4.2. Finally, VEL jointly optimizes task-execution and experience distillation under a shared policy with separate reward assignment and advantage estimation, as detailed in Section 4.3. Together, the cross-episode formulation and EGAR enable VEL to optimize experiential learning in a clear and fine-grained way. An overview of the framework is shown in Figure 2.

4.1 Cross-Episode Formulation for Isolating Experiential Learning

Cross-episode trial. To isolate experiential learning as an explicit trainable capability, VEL extends single-episode interaction into a cross-episode trial. Given a task instruction x, the agent performs at most N successive attempts on the same task instance, each corresponding to one episode. Each attempt executes the task from scratch, while the experience distilled from the pre-

ceding episode is carried over to guide the next attempt. Formally, the trial is defined as

$$\mathcal{T} = \left(\tau^0, e^0, \tau^1, e^1, \tau^2, \cdots, e^{N-2}, \tau^{N-1}\right), \tag{3}$$

where τ^n denotes the trajectory generated in the n-th attempt, and e^n the experience distilled from it to guide the next attempt. For the first attempt, no previous experience is available, and we define $e^{-1} = \varnothing$. The trial terminates when the task is solved or N attempts are reached.

Execution and distillation. VEL uses the same policy π_θ throughout the trial. In episode n, the agent acts conditioned on the task instruction and the previous experience:

$$\tau^n \sim \pi_\theta\left(\cdot \mid x, e^{n-1}\right), \qquad n = 0, \ldots, N-1. \tag{4}$$

After each attempt, the agent reflects on its trajectory and distills experience for the next attempt:

$$e^n \sim \pi_\theta\left(\cdot \mid x, \tau^n\right), \qquad n = 0, \ldots, N-2. \tag{5}$$

Thus, the policy alternates between task-execution and experience distillation across attempts.

Isolation of experiential learning. Consecutive attempts are made on the same task instance with a shared policy, and the distilled experience e^n is the only cross-episode information carried into episode $n+1$. Improvement from τ^n to τ^{n+1} can therefore be attributed more locally to the quality of e^n. As the sole information channel between attempts, e^n also forms an experience bottleneck that encourages the model to distill information supporting future task-execution, such as discovered constraints, useful subgoals, and effective strategies. Our formulation thus isolates experiential learning as an atomic capability that can be explicitly evaluated and effectively optimized.

4.2 Verifiable Experiential Learning Reward

To enable effective experiential learning, the agent should learn to reflect on prior attempts and distill high-quality experience. Therefore, VEL assigns explicit rewards to the experience distillation according to the effectiveness of the distilled experience. Under our cross-episode formulation, the effectiveness of an experience can be evaluated by whether the subsequent attempt conditioned on it achieves greater progress than the preceding episode. We therefore introduce VAS to quantify the verifiable progress achieved in each episode and define EGAR as the progress improvement guided by the distilled experience to reward the experience distillation process.

Verifiable Advancement Signal. To quantify progress beyond binary task success, VEL uses a verifier that maps the interaction trajectory to a scalar progress score. Given a task instance x and the trajectory τ^n generated in the n-th attempt, the verifier $\mathcal{V}$ computes

$$v^n = \mathcal{V}(x, \tau^n) \in [0, 1], \tag{6}$$

where v^n is the VAS of the n-th attempt. The verifier uses verifiable environment states to compute a deterministic VAS. Specific implementations are detailed in Appendix B.1.

Experience-Guided Advancement Reward. EGAR is the VAS improvement between attempts:

$$r^n_{\mathrm{EGAR}} = v^{n+1} - v^n \in [-1, 1]. \tag{7}$$

The reward is assigned to the distillation of e^n. A positive EGAR indicates that the distilled experience helps the agent advance beyond the previous attempt, while a negative EGAR indicates that the experience may mislead the next attempt. In this way, EGAR directly supervises experience distillation, encouraging the model to generate experience that advances future behavior.

4.3 Policy Optimization

Reward assignment. VEL optimizes a shared policy π_θ over two types of generated tokens: task-execution tokens produced during environment interaction, and experience-distillation tokens produced between attempts. These two types of tokens serve different roles and therefore receive different rewards. For task-execution tokens in τ^n, we assign the binary task-success reward:

$$r^n_{\mathrm{act}} = R(\tau^n). \tag{8}$$

For the experience-distillation tokens in e^n, we assign EGAR:

$$r^n_{\mathrm{exp}} = r^n_{\mathrm{EGAR}}. \tag{9}$$

Transition-Wise experience advantage estimation. We use GRPO to estimate advantages from groups of sampled cross-episode trials. For task-execution, trajectories from all attempts are flattened into the same optimization batch and optimized with group-normalized advantages computed from their binary task-success rewards. Experience distillation requires a more careful grouping strategy, as its difficulty and reward scale vary across transition positions: experience distilled after the first attempt is based on limited evidence but often leaves greater room for improvement, whereas experience from later attempts is conditioned on richer interaction histories but may have less remaining potential for advancement. Therefore, rather than normalizing EGAR across all distilled experiences, VEL computes advantages only among experiences generated at the same transition position $n \rightarrow n+1$. Given G sampled trials for the same task, the transition-wise experience advantage is computed as

$$A^n_{\text{exp},g} = \frac{r^n_{\text{EGAR},g} - \mu^n_{\text{EGAR}}}{\sigma^n_{\text{EGAR}} + \epsilon}, \qquad g = 1, \ldots, G, \tag{10}$$

where μ^n_{EGAR} and σ^n_{EGAR} are the mean and standard deviation of EGAR values within the same transition position $n \rightarrow n+1$. The resulting advantage calculations are illustrated in Figure 2.

Objective. Let $\mathcal{Y}_{\text{act}}$ denote the task-execution tokens and $\mathcal{Y}^n_{\text{exp}}$ the experience-distillation tokens at transition position $n \rightarrow n+1$. VEL optimizes the shared policy with a masked GRPO objective:

$$\max_\theta \; \mathcal{J}_{\text{VEL}}(\theta) = \mathcal{J}_{\text{GRPO}}\left(\theta; \mathcal{Y}_{\text{act}}, A_{\text{act}}\right) + \lambda \sum_{n=0}^{N-2} \mathcal{J}_{\text{GRPO}}\left(\theta; \mathcal{Y}^n_{\text{exp}}, A^n_{\text{exp}}\right), \tag{11}$$

where A_{act} is the standard GRPO advantage for task-execution, A^n_{exp} is the transition-wise EGAR advantage, and λ is the experiential-learning weight balancing the two objectives.

5 Experiments

5.1 Experimental Setup

Environments. We evaluate VEL on four long-horizon agentic benchmarks: MineSweeper (Jiang et al., 2025) for partially observable logical reasoning, Sokoban (Racanière et al., 2017) for long-horizon planning, WebShop (Yao et al., 2022) for web navigation, and ScienceWorld (Wang et al., 2022) for embodied scientific reasoning. All environments use text-based observations and actions, with environment details and prompts provided in Appendix B.2 and Appendix D, respectively.

Baselines. We compare VEL with three groups of baselines. *(i)* Prompting-based methods, including Zero-shot, ReAct (Yao et al., 2023), and Reflexion (Shinn et al., 2023), which rely on in-context reasoning or verbal reflection without parameter updates. *(ii)* RL-trained methods, including standard RL optimization methods such as PPO (Schulman et al., 2017), RLOO (Ahmadian et al., 2024), GRPO (Shao et al., 2024), and GiGPO (Feng et al., 2025), which optimize task-execution with outcome-based rewards, as well as experience-augmented RL variants such as LaMer (Jiang et al., 2025) and RetroAgent (Zhang et al., 2026a). LaMer formulates repeated attempts as a Meta-RL process, where self-reflection serves as an in-context adaptation mechanism across episodes and is optimized using the task outcome of the reflection-augmented episode. RetroAgent uses hindsight self-reflection to generate reusable textual experiences, which will be stored in a memory bank and retrieved for subsequent task-execution. Its self-reflection policy is optimized with a self-assessment reward derived from task outcomes. Both LaMer and RetroAgent jointly optimize task-execution and self-reflection through a shared policy.

Implementation details. We use Qwen3-4B (Team, 2025) as the base model for all methods. For VEL, we sample $N = 3$ episodes and set group size $G = 8$ for each task. For standard RL baselines, we set the group size to $G = 24$ to ensure a fair comparison under the same number of trajectories used for each gradient update. We set the experiential-learning weight to $\lambda = 1$ in all VEL experiments. All other implementation details and hyperparameters are kept identical across baselines and VEL for fairness, as detailed in Appendix B.3.

Table 1: Performance comparison across four benchmarks. $p@k$ denotes the task success rate (%) within k attempts. *Gain* is calculated as $p@3 - p@1$ in percentage points.

Method	MineSweeper				Sokoban				WebShop				ScienceWorld			
	$p@1$	$p@2$	$p@3$	*Gain*	$p@1$	$p@2$	$p@3$	*Gain*	$p@1$	$p@2$	$p@3$	*Gain*	$p@1$	$p@2$	$p@3$	*Gain*
Prompting																
Zero-shot	4.5	6.6	8.6	4.1	6.8	9.8	12.9	6.1	1.4	2.1	2.3	0.9	0.8	1.6	2.3	1.5
ReAct	6.3	7.0	10.9	4.6	7.2	9.6	12.5	5.3	3.1	4.5	4.5	1.4	7.8	10.2	12.5	4.7
Reflexion	5.5	7.2	9.8	4.3	6.4	9.8	12.1	5.7	2.7	3.3	3.5	0.8	7.8	12.5	17.2	9.4
RL training																
PPO	29.7	34.2	35.5	5.8	12.5	15.4	16.8	4.3	53.1	54.5	54.9	1.8	34.4	40.6	40.6	6.2
RLOO	48.8	51.2	51.6	2.8	13.5	16.6	18.8	5.3	67.6	68.4	69.1	1.5	43.8	46.9	53.1	9.4
GRPO	36.3	40.0	40.4	4.1	22.9	26.4	27.0	4.1	72.9	73.0	73.0	0.1	43.8	43.8	46.9	3.1
GiGPO	**52.0**	54.9	55.1	3.1	**41.6**	43.6	44.1	2.5	73.4	74.6	75.2	1.8	53.1	56.2	56.2	3.1
LaMer	44.1	66.4	74.4	30.3	36.7	46.1	51.6	14.9	67.8	84.4	89.1	21.3	53.9	68.0	68.8	14.9
RetroAgent	43.8	53.1	58.6	14.8	36.7	41.4	45.3	8.6	**74.2**	75.8	77.3	3.1	46.9	53.1	56.2	9.4
VEL (Ours)	47.7	**78.9**	**88.3**	**40.6**	39.8	**47.7**	**55.5**	**15.7**	70.3	**87.5**	**94.5**	**24.2**	**55.5**	**72.7**	**76.6**	**21.1**

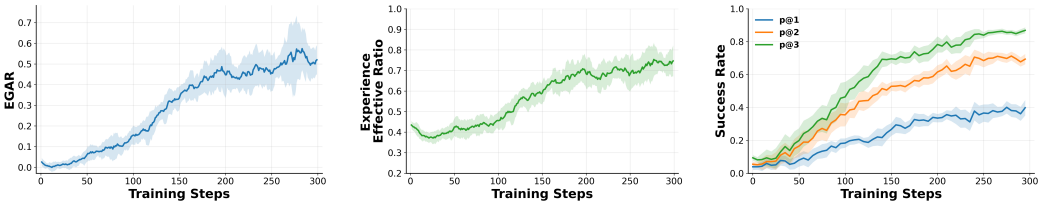

Figure 3: Training dynamics of VEL on MineSweeper: EGAR, the experience effective ratio (the fraction of distilled experiences with positive EGAR), and success rates across attempts.

Evaluation settings. We evaluate the checkpoint trained by each method through the cross-episode formulation defined in Section 4.1, allowing up to $N = 3$ attempts on each task instance. To measure performance across successive attempts, we report the cumulative task success rate $p@k$, defined as the percentage of task instances solved within k attempts.

5.2 Main Results

VEL achieves better performance. As shown in Table 1, VEL achieves the highest multi-attempt performance across all four benchmarks, outperforming both prompting and RL training baselines. Compared with methods that explicitly train self-reflection, such as LaMer and RetroAgent, VEL retains a substantial advantage. On MineSweeper, VEL achieves 88.3% $p@3$, exceeding LaMer and RetroAgent by 13.9 and 29.7 percentage points, respectively. On WebShop, VEL reaches 94.5%, compared with 89.1% for LaMer and 77.3% for RetroAgent. VEL also maintains competitive single-attempt performance despite its focus on experiential learning. Although VEL slightly trails the leading baselines at $p@1$ on some benchmarks, it still outperforms LaMer and RetroAgent on MineSweeper and Sokoban. On ScienceWorld, VEL even achieves the highest $p@1$ of 55.5%.

VEL exhibits stronger experiential learning. Beyond final success rates, the improvement from $p@1$ to later attempts directly reflects the experiential learning capability of an agent. VEL achieves the largest overall gain, measured by $p@3 - p@1$, on all four benchmarks. Specifically, VEL improves by 40.6 percentage points on MineSweeper, 15.7 on Sokoban, 24.2 on WebShop, and 21.1 on ScienceWorld. Averaged across benchmarks, VEL achieves a 49.3% relative improvement from $p@1$ to $p@3$, compared with 42.1% for LaMer and 20.4% for RetroAgent. This shows that VEL does not merely train a stronger single-attempt actor, but reinforces the ability to distill useful experience from previous attempts and reuse it in subsequent decisions. Together, these results show that VEL enables effective experiential learning and continual test-time adaptation over multiple attempts.

Table 2: Generalization performance on ALFWorld of models trained on MineSweeper under same-task and cross-task settings. *Gain* is computed as $p@5 - p@1$ in percentage points.

Method	*Same-Task* Setting						*Cross-Task* Setting
	$p@1$	$p@2$	$p@3$	$p@4$	$p@5$	*Gain*	Overall Success Rate (%)
GRPO	11.4	23.6	32.1	37.9	39.3	27.9	29.49
RetroAgent	13.6	21.4	22.9	30.0	35.0	21.4	26.92
VEL (Ours)	14.3	27.9	35.7	43.6	47.9	**33.6**	**37.18**

VEL promotes the co-evolution of experience-distillation and task-execution. Under the cross-episode formulation, improving task-execution creates a potential tension with experience-distillation. Stronger initial attempts may leave less room for further progress, making it harder to distill experience that enables additional advancement. To examine how these two processes develop during training, we track mean EGAR and the experience effective ratio, defined as the fraction of distilled experiences with positive EGAR. Surprisingly, as shown in Figure 3, both metrics continue to rise even as $p@1$ improves, supporting further gains in $p@2$ and $p@3$. In particular, on MineSweeper, the experience effective ratio rises from below 40% in the early stage of training to approximately 75%, while mean EGAR increases from nearly zero to around 0.5. These observations suggest that VEL promotes the co-evolution of experience-distillation and task-execution, reinforcing effective experience-distillation even as task-execution becomes stronger.

5.3 Generalization

We further investigate whether the experiential learning capability reinforced by VEL can generalize beyond the training environment and settings. We directly evaluate MineSweeper-trained models on ALFWorld, an unseen household instruction-following environment, without further training. We consider two evaluation settings. In the *same-task* setting, following the cross-episode formulation, the agent repeatedly attempts the same ALFWorld task instance for up to $N = 5$ attempts. In the *cross-task* setting, we extend the cross-episode formulation from repeated attempts on the same task to sequential attempts over related but distinct tasks. We construct six task sequences, each corresponding to one of ALFWorld's six predefined task categories and containing 13 distinct task instances from that category. The agent solves the tasks in each sequence in order, using the experience distilled from previous tasks to guide the next one, and we report the overall success rate.

Generalization to unseen environments. As shown in Table 2, VEL achieves the highest $p@2$ through $p@5$ on ALFWorld despite being trained only on MineSweeper. Its gain from $p@1$ to $p@5$ reaches 33.6 percentage points, compared with 27.9 for GRPO and 21.4 for RetroAgent, indicating that its experiential learning capability transfers to an unseen environment.

Generalization to longer adaptation horizons. Although trained with $N = 3$ attempts, VEL continues to benefit from additional attempts at evaluation. On ALFWorld, success rises from 35.7% at $p@3$ to 47.9% at $p@5$, adding 12.2 percentage points beyond the training horizon. This suggests that VEL can extend experience distillation and reuse to longer adaptation horizons.

Generalization to cross-task experience transfer. In the cross-task setting, VEL achieves an overall success rate of 37.18%, exceeding GRPO and RetroAgent by 7.69 and 10.26 percentage points, respectively. These results suggest that the experiential learning capability reinforced by VEL extends to cross-task settings, enabling experience transfer between related but distinct tasks.

6 Analysis

We ablate VEL's reward design and transition-wise advantage estimation and examine its performance across task horizons. Appendix C extends these analyses to the experiential-learning weight, cross-episode information transfer, and training dynamics, while Appendix E provides a Sokoban case study illustrating how distilled experience guides subsequent attempts.

6.1 Ablation Study

We ablate the two core components of VEL on MineSweeper: EGAR $r^n_{\text{EGAR}} = v^{n+1} - v^n$ (Eq. 7) and the transition-wise experience advantage (Eq. 10). For EGAR, *w/o VAS* replaces VAS with the binary task-success reward, i.e., uses $R(\tau^{n+1}) - R(\tau^n)$ as the experience reward, and *w/o Difference Term* uses the VAS of the next attempt v^{n+1} alone. *w/o Transition-Wise Advantage* normalizes EGAR jointly across all transition positions. We also report None, which keeps the cross-episode formulation but removes the experience-distillation objective, as a reference.

Table 3: Ablation study of VEL on MineSweeper.

Method	p@1	p@2	p@3	*Gain*
VEL (Ours)	47.7	78.9	88.3	**40.6**
w/o VAS	46.1	65.6	73.4	27.3
w/o Difference Term	45.3	64.1	72.7	27.4
w/o Transition-Wise Advantage	42.2	50.8	53.9	11.7
None	35.2	42.2	48.4	13.2

EGAR provides fine-grained and attributable rewards for experience-distillation. As shown in Table 3, removing VAS or the difference term reduces p@3 from 88.3% to 73.4% and 72.7% and the gain from 40.6 to about 27 percentage points. VAS credits the verifiable partial progress enabled by an experience, even before the next attempt succeeds, while the difference term credits only the advancement beyond the previous attempt. Together, they reinforce experiences that genuinely advance task-execution.

Transition-wise experience advantage calibrates the learning signal for experience-distillation. Removing the transition-wise experience advantage causes the most severe degradation among the three ablations, reducing p@3 to 53.9% and the gain to 11.7 percentage points, while p@1 decreases by only 5.5 points. This demonstrates that it is critical for effective experience-distillation. By comparing each experience only with alternatives distilled at the same transition position, it accounts for the different room for advancement across positions and provides a calibrated learning signal at every stage of the trial, which also yields consistently higher EGAR during training (Appendix C.3).

6.2 Performance Across Different Task Horizons

We estimate the horizon of each of the 128 MineSweeper evaluation tasks as the minimum number of clicks needed to reveal all safe cells from the initial state, given known mine locations. Since clicking a blank cell reveals its entire connected region, the horizon is computed via BFS over 8-connected components. We then report the p@3 of VEL, RetroAgent, and GRPO at each horizon.

VEL remains robust as the task horizon increases. As shown in Figure 4, all methods degrade as the task horizon increases, since errors accumulate over more interaction steps. Nevertheless, VEL exhibits the smallest degradation and maintains a p@3 above 75% across all horizons. In contrast, GRPO, which lacks explicit cross-episode experience distillation, remains substantially below VEL at every horizon. RetroAgent performs comparably to VEL on short-horizon tasks, but its gap to VEL widens from 1.7 at $H = 2$ to 60.4 percentage points at $H = 7$, as its outcome-derived reflection rewards give little credit to the partial advancement that EGAR captures. This suggests that VEL can reliably learn from previous attempts even when each attempt involves long interaction.

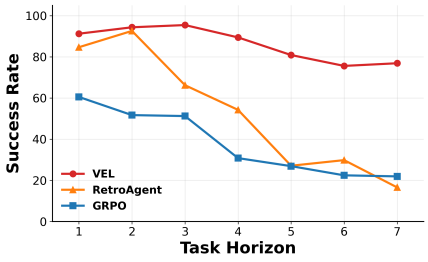

Figure 4: Success rate of VEL and baselines across different task horizons on MineSweeper.

7 Conclusion

We introduce VEL, an RL framework for verifiable experiential learning in long-horizon LLM agents. VEL formulates experiential learning as a cross-episode process and proposes EGAR with transition-wise advantage estimation, enabling a shared policy to jointly improve task-execution and experience distillation. Experiments across diverse agentic benchmarks show that VEL consistently outperforms prompting, RL, and experiential learning baselines. Further analyses demonstrate that VEL learns a transferable and scalable experiential learning capability.

AI USE STATEMENT

We used generative AI tools to assist with manuscript drafting and editing, including improving language clarity, organizing the presentation, and identifying and checking relevant literature and citations. These uses are distinct from the experimental role of LLMs in this work. In our VEL experiments, Qwen3-4B is trained as a shared policy for task execution and experience distillation: it generates actions, distills completed trajectories into textual experience, and uses that experience to guide subsequent attempts, as described in Section 4. Task success and experience-guided advancement are evaluated using benchmark outcomes and environment-derived progress measures, rather than an LLM-based judge. The corresponding model settings and prompt templates are documented in Appendices B.3 and D. The authors take responsibility for the final manuscript, the experimental methodology, and the accuracy and integrity of the reported findings.

ETHICS STATEMENT

VEL studies experiential learning in established agent benchmarks, including MineSweeper, Sokoban, WebShop, ScienceWorld, and ALFWorld. Our experiments use benchmark-provided tasks and environment feedback to evaluate adaptation across repeated attempts and related tasks, with interactions confined to the observation and action interfaces of these environments. Experience reuse nevertheless carries risks of reinforcing errors from earlier interactions or applying context-dependent strategies in situations where they are inappropriate. We therefore emphasize transparency about the experience informing agent decisions, appropriate limits on available actions, and opportunities for users to inspect, correct, or override agent behavior in downstream applications. The reported benchmark improvements do not by themselves establish suitability for deployment in open-ended settings with real users.

REPRODUCIBILITY STATEMENT

To support reproducibility, we document both VEL's learning formulation and its experimental implementation. The cross-episode process, EGAR, and joint optimization objective are defined in Section 4, with the resulting training procedure summarized in Algorithm 1. We then describe how the method is instantiated in each benchmark, including its normalized advancement signals, environment configurations, and evaluation settings (Appendices B.1 and B.2). The associated model, rollout, sampling, and optimization settings are provided in Appendix B.3, while Appendix D gives the task-execution and experience-distillation prompts. For evaluation beyond repeated attempts on the same task, Appendix D.5.1 details the ALFWorld cross-task experience-transfer protocol.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 12248–12267. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.662. URL https://doi.org/10.18653/v1/2024.acl-long.662.

Anthropic. Introducing Claude Opus 5.5, 2026. URL https://www.anthropic.com/claude-opus-5-5. Accessed: 2026-09-26.

Yuxin Chen, Yu Wang, Yi Zhang, Ziang Ye, Zhengzhou Cai, Yaorui Shi, Qi Gu, Hui Su, Xunliang Cai, Xiang Wang, An Zhang, and Tat-Seng Chua. Learning to self-verify makes language models better reasoners. *CoRR*, abs/2602.07594, 2026a. doi: 10.48550/ARXIV.2602.07594. URL https://doi.org/10.48550/arXiv.2602.07594.

Yuxin Chen, Yi Zhang, Zhengzhou Cai, Yaorui Shi, Zhiyuan Yao, Chenhang Cui, Jingnan Zheng, Yaqi Huo, Xi Su, Qi Gu, Xunliang Cai, Xiang Wang, An Zhang, and Tat-Seng Chua. Vitabench 2.0: Evaluating personalized and proactive agents in long-term user interactions. *CoRR*,

abs/2605.27141, 2026b. doi: 10.48550/ARXIV.2605.27141. URL https://doi.org/10.48550/arXiv.2605.27141.

Jingzhe Ding, Shengda Long, Changxin Pu, Huan Zhou, Hongwan Gao, Xiang Gao, Chao He, Yue Hou, Fei Hu, Zhaojian Li, Weiran Shi, Zaiyuan Wang, Daoguang Zan, Chenchen Zhang, Xiaoxu Zhang, Qizhi Chen, Xianfu Cheng, Bo Deng, Qingshui Gu, Kai Hua, Juntao Lin, Pai Liu, Mingchen Li, Xuanguang Pan, Zifan Peng, Yujia Qin, Yong Shan, Zhewen Tan, Weihao Xie, Zihan Wang, Yishuo Yuan, Jiayu Zhang, Enduo Zhao, Yunfei Zhao, He Zhu, Chenyang Zou, Ming Ding, Jianpeng Jiao, Jiaheng Liu, Minghao Liu, Qian Liu, Chongyao Tao, Jian Yang, Tong Yang, Zhaoxiang Zhang, Xinjie Chen, Wenhao Huang, and Ge Zhang. Nl2repo-bench: Towards long-horizon repository generation evaluation of coding agents. *CoRR*, abs/2512.12730, 2025. doi: 10.48550/ARXIV.2512.12730. URL https://doi.org/10.48550/arXiv.2512.12730.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. *CoRR*, abs/2505.10978, 2025. doi: 10.48550/ARXIV.2505.10978. URL https://doi.org/10.48550/arXiv.2505.10978.

Shuo He, Lang Feng, Qi Wei, Xin Cheng, Lei Feng, and Bo An. Hierarchy-of-groups policy optimization for long-horizon agentic tasks. *CoRR*, abs/2602.22817, 2026. doi: 10.48550/ARXIV.2602.22817. URL https://doi.org/10.48550/arXiv.2602.22817.

Yulun Jiang, Liangze Jiang, Damien Teney, Michael Moor, and Maria Brbic. Meta-rl induces exploration in language agents. *CoRR*, abs/2512.16848, 2025. doi: 10.48550/ARXIV.2512.16848. URL https://doi.org/10.48550/arXiv.2512.16848.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *CoRR*, abs/2503.09516, 2025. doi: 10.48550/ARXIV.2503.09516. URL https://doi.org/10.48550/arXiv.2503.09516.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tülu 3: Pushing frontiers in open language model post-training. *CoRR*, abs/2411.15124, 2024. doi: 10.48550/ARXIV.2411.15124. URL https://doi.org/10.48550/arXiv.2411.15124.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Keer Lu, Chong Chen, Bin Cui, Huang Leng, and Wentao Zhang. Pilotrl: Training language model agents via global planning-guided progressive reinforcement learning. *CoRR*, abs/2508.00344, 2025. doi: 10.48550/ARXIV.2508.00344. URL https://doi.org/10.48550/arXiv.2508.00344.

Dilxat Muhtar, Jiashun Liu, Wei Gao, Weixun Wang, Shaopan Xiong, Ju Huang, Siran Yang, Wenbo Su, Jiamang Wang, Ling Pan, and Bo Zheng. Complementary reinforcement learning. *CoRR*, abs/2603.17621, 2026. doi: 10.48550/ARXIV.2603.17621. URL https://doi.org/10.48550/arXiv.2603.17621.

OpenAI. GPT-6 Astra: A new generation of intelligence, 2026. URL https://openai.com/index/gpt-6-astra. Accessed: 2026-09-26.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory. *CoRR*, abs/2509.25140, 2025. doi: 10.48550/ARXIV.2509.25140. URL https://doi.org/10.48550/arXiv.2509.25140.

Sébastien Racanière, Theophane Weber, David P. Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, Razvan Pascanu, Peter W. Battaglia, Demis Hassabis, David Silver, and Daan Wierstra. Imagination-augmented agents for deep reinforcement learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, December 4-9, 2017*, pp. 5690–5701, 2017. URL https://dblp.org/rec/conf/nips/RacaniereWRBGRB17.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.2402.03300.

Taiwei Shi, Sihao Chen, Bowen Jiang, Linxin Song, Longqi Yang, and Jieyu Zhao. Experiential reinforcement learning. *CoRR*, abs/2602.13949, 2026a. doi: 10.48550/ARXIV.2602.13949. URL https://doi.org/10.48550/arXiv.2602.13949.

Yaorui Shi, Yuxin Chen, Zhengxi Lu, Yuchun Miao, Shugui Liu, Qi Gu, Xunliang Cai, Xiang Wang, and An Zhang. Skill1: Unified evolution of skill-augmented agents via reinforcement learning. *CoRR*, abs/2605.06130, 2026b. doi: 10.48550/ARXIV.2605.06130. URL https://doi.org/10.48550/arXiv.2605.06130.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *The Ninth International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://arxiv.org/abs/2010.03768.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *CoRR*, abs/2503.05592, 2025. doi: 10.48550/ARXIV.2503.05592. URL https://doi.org/10.48550/arXiv.2503.05592.

Xiaolong Tang, Meina Kan, Shiguang Shan, and Xilin Chen. Plan-r1: Safe and feasible trajectory planning as language modeling. *CoRR*, abs/2505.17659, 2025. doi: 10.48550/ARXIV.2505.17659. URL https://doi.org/10.48550/arXiv.2505.17659.

Qwen Team. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025. doi: 10.48550/ARXIV.2505.09388. URL https://doi.org/10.48550/arXiv.2505.09388.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=ehfRiF0R3a.

Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Panpan Xu, and Lin Lee Cheong. Reinforcement learning for self-improving agent with skill library. 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), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 1529–1550. Association for Computational Linguistics, 2026a. URL https://aclanthology.org/2026.acl-long.69/.

Ruoyao Wang, Peter A. Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.emnlp-main.775. URL https://aclanthology.org/2022.emnlp-main.775/.

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. *CoRR*, abs/2504.20073, 2025a. doi: 10.48550/ARXIV.2504.20073. URL https://doi.org/10.48550/arXiv.2504.20073.

Zixuan Wang, Yuchen Yan, Hongxing Li, Teng Pan, Dingming Li, Ruiqing Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Milestone-guided policy learning for long-horizon language agents. *CoRR*, abs/2605.06078, 2026b. doi: 10.48550/ARXIV.2605.06078. URL https://doi.org/10.48550/arXiv.2605.06078.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025b. URL https://proceedings.mlr.press/v267/wang25bx.html.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *CoRR*, abs/2602.08234, 2026. doi: 10.48550/ARXIV.2602.08234. URL https://doi.org/10.48550/arXiv.2602.08234.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/5d413e48f84dc61244b6be550f1cd8f5-Abstract-Datasets_and_Benchmarks_Track.html.

John Yang, Kilian Lieret, Jeffrey Ma, Parth Thakkar, Dmitrii Pedchenko, Sten Sootla, Emily McMilin, Pengcheng Yin, Rui Hou, Gabriel Synnaeve, Diyi Yang, and Ofir Press. Programbench: Can language models rebuild programs from scratch? *CoRR*, abs/2605.03546, 2026. doi: 10.48550/ARXIV.2605.03546. URL https://doi.org/10.48550/arXiv.2605.03546.

Shunyu Yao, Howard Chen, John Yang, and Karthik R. Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Ziang Ye, Wentao Shi, Yuxin Liu, Yu Wang, Zhengzhou Cai, Yaorui Shi, Qi Gu, Xunliang Cai, and Fuli Feng. Look before you leap: Autonomous exploration for LLM agents. *CoRR*, abs/2605.16143, 2026. doi: 10.48550/ARXIV.2605.16143. URL https://doi.org/10.48550/arXiv.2605.16143.

Mengqi Yuan, Zilong Zhou, Xinzhuang Xiong, Weiming Wu, Jiayang Sun, Jiamin Song, Kaiqian Cui, Bowen Wang, Haoyuan Wu, Yitong Li, Dunjie Lu, Haikong Lu, Qi Zhen, Xinyuan Wang, Jiaqi Deng, Yuhao Yang, Cheng Chen, Boyuan Zheng, Alex Su, Xiao Yu, Hao Zou, Saaket Agashe, Xing Han Lù, Manpreet Kaur, Zhengyang Qi, Vincent Sunn Chen, Frederic Sala, Dayiheng Liu, Junyang Lin, Zhou Yu, Yu Su, Siva Reddy, Xin Eric Wang, Peng Qi, Tianbao Xie, and Tao Yu. Osworld2.0: Benchmarking computer use agents on long-horizon real-world tasks. *CoRR*, abs/2606.29537, 2026. doi: 10.48550/ARXIV.2606.29537. URL https://doi.org/10.48550/arXiv.2606.29537.

Xiaoying Zhang, Zichen Liu, Yipeng Zhang, Xia Hu, and Wenqi Shao. Retroagent: From solving to evolving via retrospective dual intrinsic feedback. *CoRR*, abs/2603.08561, 2026a. doi: 10.48550/ARXIV.2603.08561. URL https://doi.org/10.48550/arXiv.2603.08561.

Zhiwei Zhang, Yudi Lin, Nikki Lijing Kuang, Linlin Wu, Xiaomin Li, Songtao Liu, and Fenglong Ma. Co-evolving skill generation and policy optimization. *arXiv preprint arXiv:2606.08755*, 2026b.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: LLM agents are experiential learners. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada*, pp. 19632–19642. AAAI Press, 2024. doi: 10.1609/AAAI.V38I17.29936. URL https://doi.org/10.1609/aaai.v38i17.29936.

Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, Yu Qiao, Zhaoxiang Zhang, and Jifeng Dai. Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory. *CoRR*, abs/2305.17144, 2023. doi: 10.48550/ARXIV.2305.17144. URL https://doi.org/10.48550/arXiv.2305.17144.

A LIMITATIONS

Although this work proposes a verifiable cross-episode RL framework for experiential learning and demonstrates its effectiveness across multiple agent benchmarks, it still has several limitations. First, VEL is trained primarily under a same-task cross-episode formulation, which provides a cleaner setting for credit assignment. However, distilling reusable experience across tasks is a harder and important problem. We adopt same-task training as a controlled formulation and evaluate cross-task transfer only as an initial step toward more general experiential learning. Second, VEL requires multiple attempts during training, which inevitably incurs a larger budget than standard RL. In our experiments, we control the trajectory budget when comparing against RL baselines and allow the agent to stop once the task is solved. Nonetheless, improving the sample efficiency of cross-episode training and reducing the cost of multi-attempt optimization remain important directions for future work. Third, due to computational constraints, our experiments are limited to moderate-size LLMs and a set of representative agentic benchmarks. We do not yet explore whether the benefits of VEL continue to scale to substantially larger models, longer-horizon environments, or more open-ended real-world agent tasks. While our results suggest that verifiable experiential learning is a promising direction, further scaling studies are needed to understand its behavior under stronger base models and more challenging agentic settings.

B IMPLEMENTATION AND EXPERIMENTAL DETAILS

B.1 VERIFIABLE ADVANCEMENT SIGNAL

Definition and Design Principles of VAS. The "V" in VAS stands for *Verifiable*. Its primary purpose is to provide, under the cross-episode experiential learning setting, an environment-derived statistic for each attempt that characterizes the degree of task progress, and to use changes in this statistic to measure whether the distilled experience truly enables subsequent attempts to achieve additional progress. Relying solely on the binary final task outcome would discard substantial intermediate progress: even if an attempt does not successfully complete the task, it may have already accomplished several critical steps, eliminated incorrect branches, or substantially reduced the distance to the goal state. If such intermediate progress cannot be reliably captured, experiential learning cannot obtain fine-grained and attributable supervision.

Accordingly, the design of VAS follows three principles. First, it must be derived from objective and well-defined environment signals, such as environment-provided scores, rewards, state changes, or quantities that can be reproducibly computed from environment states, rather than from LLM-as-a-judge evaluations or human annotations. Second, it must remain comparable across different attempts, such that the difference between two consecutive attempts can be interpreted as genuine additional progress. Third, it should preserve as much fine-grained structure of the task process as possible, so that experiential learning focuses not only on whether the final task is successfully completed, but also on whether subsequent attempts advance further and make more effective decisions in the environment. Following these principles, we first compute a benchmark-specific raw progress measure q^n for each attempt and then map it to a normalized VAS $v^n \in [0, 1]$. The difference between consecutive normalized VAS values, $v^{n+1} - v^n$, is used as the supervision signal for the experience distillation stage.

VAS is explicitly instantiated only for benchmarks that participate in experiential learning training. For ALFWorld, which is used solely for evaluation and does not participate in this training process, we do not additionally construct a training-time VAS signal.

MineSweeper. MineSweeper is a partially observable logical reasoning task in which the agent must infer hidden mines from numerical clues on the board and progressively reveal safe cells. In this benchmark, task progress is measured by the verifiable advancement accumulated within the environment during an attempt. Specifically, the environment's step reward is directly tied to verifiable changes in the board state. Revealing a single safe cell yields a reward of 0.5; triggering a cascading expansion of a blank region and thereby revealing multiple cells in a single step yields a reward of 2.0; completing the board in that step yields a reward of 10.0 and terminates the game; revealing a mine yields a reward of -0.1 and immediately terminates the attempt; and an action

that produces no valid board-state change, such as repeatedly selecting an already known cell or outputting an invalid coordinate, receives a reward of -1.0.

Let q^n_{MS} denote the cumulative environment reward over the task-execution trajectory of the n-th attempt. Under the default configuration of at most seven decision turns, the theoretical lower and upper bounds are -7 and 22, respectively. The lower bound corresponds to seven ineffective actions, while the upper bound corresponds to six non-terminal cascading rewards of 2.0 followed by a successful final action with reward 10.0. We therefore normalize the attempt-level progress as

$$v^n_{\mathrm{MS}} = \frac{q^n_{\mathrm{MS}} + 7}{29}.$$

This fixed affine transformation preserves the ordering and the differences of the original environment progress while placing the VAS on a common scale. The corresponding improvement between consecutive attempts is $v^{n+1}_{\mathrm{MS}} - v^n_{\mathrm{MS}}$.

Sokoban. Sokoban is a long-horizon planning task in which the agent must push boxes to target locations within a finite grid while avoiding irreversible deadlock states. In this benchmark, progress is measured by the remaining shortest-solution distance from the current state to the goal state. Specifically, the environment first computes the length of a shortest solution from the initial state, denoted by d_0. During interaction, after each action, it recomputes the length of the shortest remaining action sequence from the current state to the goal state. If the task has already been solved, the remaining distance is set to zero. If the agent enters an unsolvable or deadlocked state, its remaining distance is assigned the initial solution distance d_0, corresponding to zero effective progress.

For the n-th attempt, let d^n_{end} denote the remaining solution distance at the end of the attempt. In the default setting, the initial shortest solution contains at most 21 atomic actions. We therefore define the normalized VAS as

$$v^n_{\mathrm{Sok}} = \mathrm{clip}\left(\frac{d_0 - d^n_{\mathrm{end}}}{21}, 0, 1\right).$$

The clipping handles states whose remaining distance is larger than the initial distance, while a deadlocked state receives zero progress because its remaining distance is reset to d_0. A subsequent attempt receives a higher VAS when it terminates closer to the goal state. The experience-distillation reward is computed from the difference between the normalized VAS values of two consecutive attempts. The remaining solution distance is computed using a BFS-based shortest-path search over the environment state space, providing a deterministic and reproducible environment-side progress measure.

WebShop. WebShop is an interactive web-shopping benchmark in which the agent must search, filter, and identify a target product across multiple product pages according to natural-language shopping requirements. In this benchmark, VAS measures environment-verifiable intermediate progress during interaction rather than relying directly on whether the final purchase is successfully completed. The environment tracks a sequence of milestones corresponding to reaching a valid search result, opening a valid product-detail page, selecting required product options when they exist, and completing the purchase. Each milestone can be completed at most once, and repeated visits do not increase the count.

Let q^n_{Web} denote the number of milestones completed during the task-execution trajectory of the n-th attempt. The maximum number of milestones is task-dependent: it is 3 for tasks without option requirements and 4 for tasks with option requirements. Equivalently, if M_x denotes the maximum number of milestones for task x, then $M_x = 3 + \mathbf{1}[\text{task } x \text{ has option requirements}]$. We define

$$v^n_{\mathrm{Web}} = \frac{q^n_{\mathrm{Web}}}{M_x}.$$

Thus, the difference between consecutive normalized values measures the additional verifiable browsing progress achieved by the subsequent attempt. Since the signal is derived entirely from observable page states and transitions, it remains objective and reproducible across attempts.

ScienceWorld. ScienceWorld is a text-based environment for scientific experimentation and multi-step reasoning, where the agent must observe, manipulate, and verify objects to progressively

complete a given scientific task. In this benchmark, VAS is instantiated using the environment's native task-completion score to characterize intermediate progress, rather than relying on the binary final-outcome reward used for training. Specifically, during interaction, the environment returns a task-completion score in the range from 0 to 100 that reflects the degree of task progress represented by the current state. Let q^n_{Sci} denote the highest score reached during the n-th attempt. We normalize this score as

$$v^n_{\mathrm{Sci}} = \frac{q^n_{\mathrm{Sci}}}{100} = \frac{\max_t s^n_t}{100}.$$

This design preserves intermediate progress achieved through correctly completed experimental steps or subgoals, even when the attempt does not ultimately solve the task. The difference between consecutive normalized scores is used as the experience-guided advancement reward.

Summary. Although the specific instantiations of VAS differ across the four benchmarks used for experiential learning training, they follow the same underlying principle: MineSweeper uses normalized cumulative environment reward, Sokoban uses normalized reduction in remaining solution distance, WebShop uses the fraction of task-specific milestones completed, and ScienceWorld uses the normalized maximum task-completion score returned by the environment. In all four benchmarks, the normalized VAS lies in $[0, 1]$, and the difference between consecutive attempts provides the experience-guided advancement signal. ALFWorld is used only for evaluation in this work and therefore does not require a separate training-time VAS instantiation.

B.2 Environment and Evaluation Setup

MineSweeper. MineSweeper is a partially observable logical reasoning task in which the agent must infer hidden mines from numerical clues on revealed cells and progressively clear all safe cells (Jiang et al., 2025). In our experiments, we use a 6×6 board with 3 hidden mines and render the board state directly as a textual grid with row and column indices. At each decision turn, the agent can reveal only one unopened cell, with the action specified by its coordinate. The environment provides feedback through numerical clues, cascading expansion of blank regions, or a termination signal when a mine is revealed. Each attempt contains at most 7 decision turns. For the same task instance, the initial board configuration remains fixed across repeated attempts, allowing information discovered in previous attempts, such as safe regions, high-risk areas, and incorrect inferences, to be distilled into experience that can be reused in subsequent attempts.

Sokoban. Sokoban is a classic long-horizon planning and deadlock-avoidance task in which the agent must push boxes to target locations within a finite grid while avoiding irreversible dead-end states (Racanière et al., 2017). In our setting, the environment is represented as a textual grid with row indices, explicitly marking walls, floor cells, target locations, boxes, and the player position. We use a compact 6×6 room with 2 boxes. The agent's primitive action space is {`up`, `down`, `left`, `right`}, while each decision turn requires the agent to output 3 consecutive atomic actions. Therefore, each attempt is limited to at most 7 decision turns, corresponding to a budget of 21 atomic actions. This setting requires the model not only to determine whether local actions are valid, but also to plan the relative positions of the player and boxes within a limited interaction budget and proactively avoid moves that would lead to unsolvable states.

WebShop. WebShop is an interactive benchmark designed around realistic web shopping scenarios, where the agent must follow natural-language shopping instructions, navigate across search pages, result pages, and product-detail pages, and ultimately identify a product that satisfies the specified constraints (Yao et al., 2022). In our setting, we use a unified text-only web interface: the model receives the textual content of the current page together with the corresponding admissible actions, and progresses through the task via actions such as searching, clicking, and page navigation. Unlike grid-based planning tasks, the main challenges in WebShop arise from compositional product constraints, noisy information in webpage text, and the need to balance retrieval and browsing strategies. Under the repeated-attempt setting, the same shopping objective remains fixed across attempts. As a result, information revealed by previous failures, such as keyword choices, attribute conflicts, and promising candidate products, can be distilled and reused in subsequent attempts to support more effective retrieval and filtering strategies.

ScienceWorld. ScienceWorld is a text-based environment for scientific experimentation and procedural reasoning, covering a range of elementary science activities including observation, manipulation, verification, and experimental interpretation (Wang et al., 2022). In our setting, each task instance is presented to the model through textual observations together with a list of admissible actions, where actions are typically composed of a verb and one or more environment objects. The environment also provides the task description and the current task-completion score. We use the *easy simplifications* preset and limit the maximum interaction length of each attempt to 30 steps. Compared with WebShop, ScienceWorld places greater emphasis on state tracking and causal reasoning throughout multi-step experimental procedures. As a result, experience distilled from previous attempts regarding key objects, experimental order, and ineffective actions can often directly improve the completion progress of subsequent attempts.

ALFWorld. ALFWorld reformulates embodied household tasks as a text-based interactive environment, covering a range of everyday activities such as picking and placing objects, cleaning, heating, and cooling (Shridhar et al., 2021). In this work, ALFWorld is used solely for evaluation and does not participate in experiential learning training. It serves to evaluate whether the experiential learning capability acquired on MineSweeper can transfer to an unseen environment. We use its text-based interface consisting of textual observations and a list of admissible actions. Under the same-task setting, the model repeatedly attempts the same task instance, whereas under the cross-task setting, experience distilled from one task is transferred to a subsequent task of the same type but with a different instance. Specifically, we organize the tasks into 6 task-type sequences, each containing 13 related task instances, to evaluate whether the model can distill transferable task-type experience from specific instances rather than merely memorizing object locations or action sequences within a single scene.

B.3 Training Details and Hyperparameters

B.3.1 Training Procedure and Credit Assignment

Overall Training Setup. All experiments initialize the policy from Qwen3-4B with thinking mode disabled. Training is conducted on a single node with 8 GPUs. We use vLLM as the rollout backend and GRPO for policy optimization. Consistent with the cross-episode formulation introduced in the main text, each rollout for a task consists of 3 consecutive attempts. The first attempt starts directly from the initial state. After each unsuccessful non-final attempt, the model enters an experience distillation stage, where it distills the completed trajectory into a concise textual experience that is provided to the subsequent attempt. We adopt an experience-only setting throughout our main experiments. Only the distilled experience from the immediately preceding attempt, rather than the full historical trajectory, is transferred across attempts. The benchmarks used for experiential learning training are MineSweeper, Sokoban, ScienceWorld, and WebShop. ALFWorld is used solely for evaluation and does not participate in this training process.

Optimization and Credit Assignment. VEL jointly optimizes two types of generations, namely play and experience distillation, within a unified GRPO framework, while assigning them different supervision signals. For task execution, each attempt receives a binary task-success reward: 1 if the task is completed successfully and 0 otherwise. Fine-grained environment rewards and progress measures are used to construct VAS and EGAR, rather than being directly assigned as task-execution rewards. For the experience distillation stage, rather than propagating the play return of the subsequent attempt back to the distilled experience through cross-attempt discounting, we use the additional verifiable environment-side progress achieved between two consecutive attempts as the supervision signal, corresponding to EGAR in the main text. Here, "progress" is always defined by objective signals from the environment. The benchmark-specific raw progress measure is first mapped to the normalized VAS defined in Section B.1, and the difference between consecutive normalized VAS values is used as EGAR. Consequently, the reward assigned to the experience distillation stage no longer depends solely on whether the subsequent attempt ultimately succeeds, but instead on whether the distilled experience actually enables the agent to achieve additional progress.

To avoid interference between play and experience-distillation samples caused by differences in reward scale and semantics, play samples retain task-level grouping, whereas experience-distillation samples are regrouped according to their cross-attempt transition positions and normalized inde-

Algorithm 1 VEL Training Framework

Require: Task distribution $\mathcal{D}$, shared policy π_θ, attempt budget N, group size G, experiential-learning weight λ
Ensure: Updated policy π_θ

for each training step **do**
  Sample a minibatch $\mathcal{B} \subset \mathcal{D}$ and initialize $\mathcal{Y}_{\text{act}} \leftarrow \varnothing$ and $\mathcal{Y}^n_{\text{exp}} \leftarrow \varnothing$ for all n
  1. Cross-Episode Rollout
  for each task $x \in \mathcal{B}$ **do**
    for each group member $g \in \{1, \dots, G\}$ **do**
      Set $E_g^{-1} \leftarrow \varnothing$
      for attempt index $n = 0, \dots, N-1$ **do**
        Reset the environment to the initial state of task instance x
        Sample task trajectory $\tau_g^n \sim \pi_\theta(\cdot \mid x, E_g^{n-1})$
        Compute $r^n_{\text{act},g} = R(\tau_g^n)$ and $v_g^n = \mathcal{V}(x, \tau_g^n)$
        Add active task-execution tokens to $\mathcal{Y}_{\text{act}}$
        if τ_g^n solves x **then**
          Mark member g as terminated and **break** the current trial
        else if $n < N-1$ **then**
          Sample distilled experience $e_g^n \sim \pi_\theta(\cdot \mid x, \tau_g^n)$
          Set the next-attempt context $E_g^n \leftarrow e_g^n$
        end if
      end for
    end for
  end for
  2. Task-Execution Advantage
  Compute task-execution GRPO advantages A_{act} within the task-level rollout groups
  3. Transition-Wise Experience Advantage
  for each task $x \in \mathcal{B}$ **do**
    for each transition position $n = 0, \dots, N-2$ **do**
      Define $\mathcal{G}_n(x) \leftarrow \{g : e_g^n$ was generated and τ_g^{n+1} was executed$\}$
      if $\mathcal{G}_n(x) \neq \varnothing$ **then**
        Compute $r^n_{\text{exp},g} \leftarrow v_g^{n+1} - v_g^n$ for each $g \in \mathcal{G}_n(x)$
        Normalize rewards within $\mathcal{G}_n(x)$ and construct $\mathcal{Y}^n_{\text{exp}}$ from $\{e_g^n : g \in \mathcal{G}_n(x)\}$
      end if
    end for
  end for
  4. Joint Policy Optimization
  Rescale A^n_{exp} by λ and update π_θ with a masked GRPO step over $\mathcal{Y}_{\text{act}}$ and $\{\mathcal{Y}^n_{\text{exp}}\}_{n=0}^{N-2}$
  if the validation interval is reached **then**
    Run validation and checkpointing
  end if
end for

pendently within each group when computing GRPO advantages. Unless otherwise specified, the experiential-learning weight λ is fixed to 1. Algorithm 1 summarizes the VEL training procedure, including repeated-attempt rollouts, reward assignment, and policy optimization.

For a group member that solves the task at attempt n, the current trial terminates immediately: no later task-execution or experience-distillation tokens are collected for that member. In the batched implementation, the same behavior is realized with the success indicator and active masks. A transition position $n \to n+1$ contributes to experience-distillation training only when both e_g^n and τ_g^{n+1} exist; consequently, early-terminated members are excluded from $\mathcal{G}_n(x)$.

B.3.2 Sampling and Validation

At each parameter update, we sample a minibatch from the training split of the corresponding benchmark and generate a group of repeated-attempt rollouts in parallel for the tasks in the batch. We then compute rewards and advantages for the task-execution and experience-distillation samples collected from these rollouts, followed by a single GRPO update. All experiments perform an initial validation before training and conduct periodic validation and checkpointing throughout the training process. During validation, we use stochastic sampling with a temperature of 0.7, top-p of 0.8, and

Table 4: Shared numerical hyperparameters used by VEL across all training benchmarks.

Hyperparameter	Value	Description
Training Configuration		
Number of attempts N	3	Number of repeated attempts for each task instance.
Group size G	8	Number of rollout groups sampled for each task.
Optimization and Regularization		
Learning rate	1×10^{-6}	Learning rate for policy optimization.
Policy-update mini-batch size	64	Mini-batch size used during GRPO policy updates.
Optimization epochs per update	1	Number of optimization epochs per GRPO policy update.
Weight decay	0.01	Weight-decay coefficient used by the optimizer.
Gradient clipping	1.0	Maximum gradient norm used for clipping.
Experiential-learning weight λ	1.0	Weight assigned to the experience-distillation objective.
Sampling Configuration		
Training temperature	1.0	Sampling temperature used during training rollouts.
Training top-p	1.0	Nucleus-sampling threshold used during training.
Training top-k	-1	Top-k filtering disabled during training.
Validation Configuration		
Validation batch size	128	Batch size used during validation.
Validation temperature	0.7	Sampling temperature used during validation.
Validation top-p	0.8	Nucleus-sampling threshold used during validation.
Validation top-k	20	Top-k sampling parameter used during validation.
Validation frequency	5	Number of training steps between consecutive validations.

Table 5: Benchmark-specific training configurations used by VEL.

Benchmark	Train Batch	Micro / GPU	Prompt Length	Response Length	Max Steps	Max Turns	Epochs
MineSweeper	16	8	2048	1024	15	7	300
Sokoban	16	8	2048	1024	30	7	300
WebShop	8	16	8192	1024	40	12	150
ScienceWorld	8	4	8192	512	30	30	200

top-k of 20. We additionally fix the random seed to 20 to reduce variance across different validation rounds.

B.3.3 Shared and Benchmark-Specific Configurations

Training Configuration and Benchmark-Specific Differences. We use AdamW for optimization and GRPO for policy updates, with vLLM as the rollout backend. Across all training benchmarks, VEL adopts an experience-only cross-attempt configuration, where each later attempt is conditioned on the distilled experience from the immediately preceding attempt. Both KL loss and KL-in-reward are disabled. Table 4 summarizes the shared numerical hyperparameters. We perform an initial validation before training and subsequently validate every five training steps.

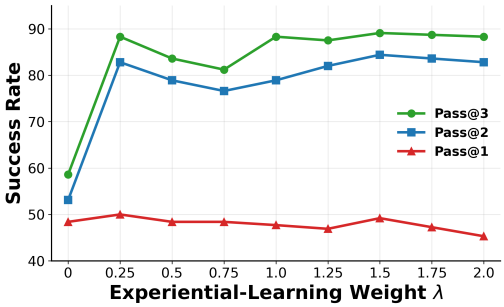

Figure 5: Success rate of VEL trained with different experiential-learning weights λ on MineSweeper.

Table 5 reports the benchmark-specific configurations for the four training environments. The main differences across benchmarks lie in training batch size, per-GPU micro-batch size, context and response-length budgets, interaction horizon, and the number of training epochs. MineSweeper and Sokoban involve relatively compact observation and action contexts and therefore use shorter prompt-length budgets together with a training batch size of 16 and 300 training epochs. In contrast, ScienceWorld must accommodate longer task descriptions, interaction histories, and admissible-action lists, and thus uses a larger input-length budget together with a reduced per-GPU micro-batch size. For WebShop, we use the benchmark-specific training configuration listed in Table 5. Since ALFWorld is used solely for evaluation in our experiments, it is not included in the training configuration table.

C Additional Analysis

Beyond implementation details, we provide three additional analyses that examine the sensitivity to the experiential-learning weight, the choice of information transferred across consecutive attempts, and the training dynamics of transition-wise advantage estimation.

C.1 Influence of Experience-Learning Weight

We study the influence of the experiential-learning weight λ, which balances task-execution optimization and experience-distillation optimization in Eq. 11. As shown in Figure 5, setting $\lambda = 0$, which removes the experience-distillation objective, leads to limited multi-attempt improvement: $p@2$ and $p@3$ remain much closer to $p@1$. In contrast, introducing a positive experiential-learning weight substantially improves later-attempt performance. Even a small value such as $\lambda = 0.25$ yields a large increase in both $p@2$ and $p@3$, indicating that directly optimizing experience distillation is crucial for enabling the agent to benefit from previous attempts.

VEL is also robust to the precise choice of λ within a broad range. For $\lambda \in [0.25, 2.0]$, $p@3$ consistently remains high, roughly between 81% and 89%, while $p@2$ also stays substantially above the $\lambda = 0$ setting. Meanwhile, $p@1$ changes only mildly across different values of λ, suggesting that strengthening experiential learning does not significantly compromise the single-attempt task-execution capability. These results show that VEL does not rely on delicate tuning of the experience-learning weight.

C.2 Information Transferred across Episodes

We further study what information should be transferred across consecutive attempts. As shown in Table 6, removing cross-episode information largely weakens multi-attempt improvement, the gain from $p@1$ to $p@3$ drops to only 9.3 points. Providing the full previous trajectory improves over no experience, achieving 20.3 points gain, indicating that past interactions indeed contain useful information for subsequent attempts. However, it remains far below distilled experience, which achieves 88.3% $p@3$ and 40.6 points gain. This suggests that directly exposing the agent to the full trajectory is not an effective substitute for experience distillation. Raw trajectories are often long and contain redundant observations, failed actions, and incidental details, which may distract

Table 6: Ablation on the information transferred across consecutive attempts.

Cross-Episode Information	$p@1$	$p@2$	$p@3$	*Gain*
No Experience	47.7	53.9	57.0	9.3
Full Trajectory	47.7	61.0	68.0	20.3
Distilled Experience (Ours)	47.7	78.9	88.3	**40.6**

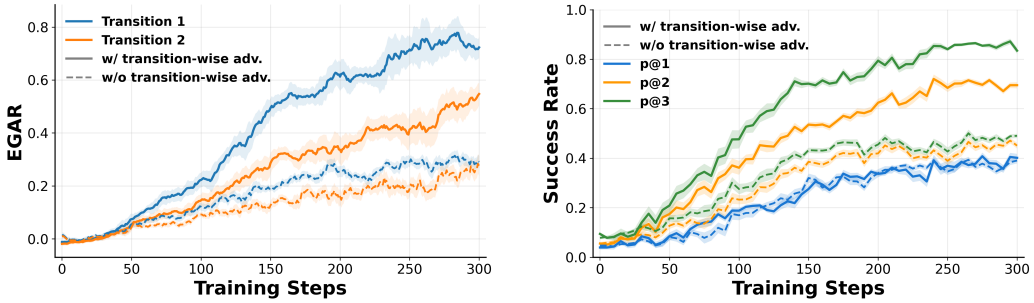

Figure 6: Training dynamics of VEL w/ and w/o transition-wise advantage estimation on MineSweeper. Left: EGAR at each transition position. Right: success rates across attempts.

the model or induce context drift during subsequent decision making. In contrast, the distilled experience produced by VEL serves as a compact and learned information bottleneck, preserving key constraints, useful discoveries, and actionable strategies from prior attempts while filtering out irrelevant or misleading details. These results support the design of using distilled textual experience as the sole cross-episode information carrier.

C.3 Training Dynamics of Transition-Wise Advantage Estimation

Figure 6 compares the training dynamics of VEL with and without transition-wise advantage estimation on MineSweeper, where the latter normalizes EGAR jointly across transition positions. With transition-wise estimation, EGAR at both transition positions rises above 0.5 by the end of training, whereas joint normalization keeps it around 0.3. Correspondingly, $p@3$ exceeds 80% with transition-wise estimation but plateaus around 50% under joint normalization, while the difference in $p@1$ is much smaller. This indicates that transition-wise estimation mainly improves experience-distillation rather than single-attempt task-execution.

D Prompt Templates

We report the prompt templates used by VEL for task execution and cross-attempt experience distillation. For same-task rollouts, each task instance is attempted up to three times on MineSweeper, Sokoban, WebShop, and ScienceWorld, and up to five times in ALFWorld evaluation. In these same-task settings, the first attempt is performed without any prior distilled experience. After each unsuccessful non-final attempt, the policy distills the completed trajectory into a concise natural-language experience that conditions the subsequent attempt. Unless otherwise specified, the experiments in this paper adopt the `experience-only` setting, in which each subsequent attempt receives only the distilled experience from the immediately preceding attempt rather than the full historical trajectory.

Shared conventions. All execution prompts instruct the model to reason before acting and to place the final action inside `<action>` and `</action>` tags. Experience distillation prompts ask the model to analyze failure or inefficiency and to end with a concise experience summary inside `<remark>` and `</remark>` tags. The placeholders shown below, such as {`init_observation`}, {`current_trajectory`}, and {`experience`}, denote benchmark-specific observations, trajectories, and experience summaries that are instantiated during interaction.

D.1 MINESWEEPER

MineSweeper uses a board-centric prompt that explicitly defines the cell semantics, the reveal rules, and the coordinate-based action format. Below we report the execution and experience distillation templates verbatim. Under the main `experience-only` configuration, later attempts receive only the previous experience summaries rather than the raw historical trajectories.

MineSweeper Execution Prompt

```
You are an expert agent operating in the Minesweeper game.
You will be given a two dimensional {board_size} by {board_size} board,
     with {n_mines} hidden mines.
The rows and columns are indexed from 1 to {board_size}.

# Cell States
- Unopened cells (?): cells that are yet to be revealed and may contain
     a mine.
- Blank cells (.): opened and non-mine cells, and they have no
    neighboring mines
- Numbered cells (1-8): opened and non-mine cells, and the number
    indicates how many mines are in the eight neighboring cells,
    including those diagonally adjacent. For example, a cell with a '8'
     means all its neighboring cells contain mines.
- Mine cells (*): opened cells that contain a mine.

# Your Goal
Your goal is to clear the board by revealing all the cells that don't
    contain mines, without detonating any of the hidden mines scattered
     throughout the board.
Use clues about the number of neighboring mines in each field to reason
     about the position of mines and non-mine cells.

# Reveal Rules
Your admissible action is to choose ONE unopened cell (?) to reveal per
     turn. The outcome depends on the content of that cell:
- Blank cell (.): That cell is revealed, and all contiguous blank cells
     plus their bordering numbered cells are automatically revealed (
    auto-cascade).
- Numbered cell (1--8): Only that single cell is revealed, showing the
    count of neighboring mines.
- Mine (*): The game ends immediately in a loss.

# Observation
The initial state of the game is:
{init_observation}{experience}{current_trajectory}

Now it's your turn to make a move.
- You should first reason step-by-step about the current situation ---
    observe the status of the board, inferring the states of unopened
    cells (?).
- Then choose ONE unopened cell (?) to reveal. Put the index of cell in
     the format of "(row, col)" within the <action> </action> tag.
```

MineSweeper Experience Distillation Prompt

```
You are an expert agent operating in the Minesweeper game.
You will be given a two dimensional {board_size} by {board_size} board,
     with {n_mines} hidden mines.
The rows and columns are indexed from 1 to {board_size}

# Cell States
```

```
- Unopened cells (?): cells that are yet to be revealed and may contain
    a mine.
- Blank cells (.): opened and non-mine cells, and they have no
   neighboring mines
- Numbered cells (1-8): opened and non-mine cells, and the number
   indicates how many mines are in the eight neighboring cells,
   including those diagonally adjacent. For example, a cell with a '8'
    means all its neighboring cells contain mines.
- Mine cells (*): opened cells that contain a mine.

# Your Goal
Your goal is to clear the board by revealing all the cells that don't
   contain mines, without detonating any of the hidden mines scattered
    throughout the board.
Use clues about the number of neighboring mines in each field to reason
    about the position of mines and non-mine cells.

# Reveal Rules
Your admissible action is to choose ONE unopened cell (?) to reveal per
    turn. The outcome depends on the content of that cell:
- Blank cell (.): That cell is revealed, and all contiguous blank cells
    plus their bordering numbered cells are automatically revealed (
   auto-cascade).
- Numbered cell (1--8): Only that single cell is revealed, showing the
   count of neighboring mines.
- Mine (*): The game ends immediately in a loss.

# Your Task
You will be given the history of a past attempt.
Your job now is to **analyze the past attempt**, identify any **
   mistakes or inefficiencies**, and distill a **concise experience
   with an improved plan** for your next try starting from the
   original initial state.

# Past Attempt
The initial state of the game is:
{init_observation}{current_trajectory}
The task is NOT successfully completed.

Now it's your turn to analyze the past attempt, distill experience from
    it, and come up with a new plan of action.

- Your response should first be step-by-step reasoning about the
   strategy and path you took to attempt to complete the task.
   Identify where things went wrong or could be better.
- Then devise a concise, new plan of action that accounts for your
   mistake with reference to specific actions that you should have
   taken.
- Finally, end the response with the distilled experience, including
   the identified mistake and improved plan, inside <remark> </remark>
    tags, to guide the next trial.
```

D.2 SOKOBAN

Sokoban uses a structured symbolic grid description together with a multi-action decision format. In each decision round, the agent outputs three atomic actions, which are then executed sequentially in the environment. Below we report the execution and experience distillation templates verbatim.

Sokoban Execution Prompt

```
You are an expert agent operating in the Sokoban environment.

# Symbols and Their Meaning
- Walls (‘#‘): These block movement. You can’t move through or push
    anything into walls.
- Floor (‘_‘): Open spaces where you can walk and move boxes.
- Targets (‘O‘): The spots where boxes need to go.
- Boxes (‘X‘): These are what you need to push onto the targets.
- Player (‘P‘): That’s you! You’ll move around the grid to push boxes.
- Box on Target (‘√‘): A box successfully placed on a target.
- Player on Target (S): You standing on a target.

# Goal
Your goal is to push all the boxes (‘X‘) onto the target spots (‘O‘).
    Once all boxes are on the targets, you win!

# Rules
Your admissible actions are ["up", "down", "left", "right"].
You can only push one box at a time. You can’t pull boxes, so plan
    ahead to avoid getting stuck.
You can’t walk through or push boxes into walls (‘#‘) or other boxes.
To avoid traps, do not push boxes into corners or against walls where
    they can’t be moved again.

# Observations
The initial state of the game is:
{init_observation}{experience}{current_trajectory}

Now it’s your turn to make moves (choose the next {num_actions_per_turn
    } actions).

- Your response should first be step-by-step reasoning about the
    current situation ---observe the positions of boxes and targets,
    plan a path to push a box toward a target, and avoid traps like
    corners or walls.
- Then choose {num_actions_per_turn} admissible actions and present
    them within <action> </action> tags (separated by comma).
```

Sokoban Experience Distillation Prompt

```
You are an expert agent operating in the Sokoban environment.

# Symbols and Their Meaning
- Walls (‘#‘): These block movement. You can’t move through or push
    anything into walls.
- Floor (‘_‘): Open spaces where you can walk and move boxes.
- Targets (‘O‘): The spots where boxes need to go.
- Boxes (‘X‘): These are what you need to push onto the targets.
- Player (‘P‘): That’s you! You’ll move around the grid to push boxes.
- Box on Target (‘√‘): A box successfully placed on a target.
- Player on Target (S): You standing on a target.

# Your Goal
Your goal is to push all the boxes (‘X‘) onto the target spots (‘O‘).
    Once all boxes are on the targets, you win!

# Rules
Your admissible actions are ["up", "down", "left", "right"].
You can only push one box at a time. You can’t pull boxes, so plan
    ahead to avoid getting stuck.
```

```
You can't walk through or push boxes into walls ('#') or other boxes.
To avoid traps, do not push boxes into corners or against walls where
    they can't be moved again.

# Your Task
You will be given the history of a past attempt.
Your job is to **analyze the past sequence**, identify any **mistakes
    or inefficiencies**, and then distill a **concise experience with
    an improved plan** starting from the original initial state.

# Past Attempt
The initial state of the game is:
{init_observation}
{current_trajectory}
The task is NOT successfully completed.

Now it's your turn to analyze the past attempt, distill experience from
     it, and come up with a new plan of action.

- Your response should first be step-by-step reasoning about the
    strategy and path you took to attempt to complete the task.
    Identify where things went wrong or could be better.
- Then devise a concise, new plan of action that accounts for your
    mistake with reference to specific actions that you should have
    taken.
- Finally, end the response with the distilled experience and improved
    plan inside <remark> </remark> tags, to guide the next trial.
```

D.3 WEBSHOP

WebShop uses a text-rendered webpage description together with a dynamic page-dependent action list. The prompts below specify both task execution and post-attempt experience distillation for this interface. They emphasize search-query construction, navigation across result and product pages, and attribute-aware candidate selection under noisy textual observations. In the main `experience-only` setting, later attempts are conditioned on the distilled experience summaries from previous attempts rather than the full raw browsing history.

WebShop Execution Prompt

```
You are an expert autonomous agent operating in the WebShop e-commerce
    environment.
Your task is to: {task_description}.

# Interface
You interact with a text-rendered shopping website. The admissible
    action set is page-dependent, but actions follow the WebShop text
    interface, e.g., search[<your query>] and click[<target>].

# Observation
{init_observation}
{experience}
{current_trajectory}

Your admissible actions of the current situation are:
[
{available_actions}
].

Now it's your turn to take one action for the current step.
- Your response should first be step-by-step reasoning about the
    current situation.
```

```
- Then think carefully about which admissible action best advances the
    shopping goal.
- Once you've finished your reasoning, you should choose an admissible
    action for current step and present it within <action> </action>
    tags.
```

WebShop Experience Distillation Prompt

```
You are an expert autonomous agent operating in the WebShop e-commerce
    environment.
Your task is to: {task_description}.

# Interface
You interact with a text-rendered shopping website. The admissible
    action set is page-dependent, but actions follow the WebShop text
    interface, e.g., search[<your query>] and click[<target>].

# Your Task
You will be given the history of a past attempt.
Your job is to analyze the past sequence, identify any mistakes or
    inefficiencies, and then distill a concise experience with an
    improved plan starting from the original initial state.

# Past Attempt
{init_observation}
{current_trajectory}
The task is NOT successfully completed.

Now it's your turn to analyze the past attempt, distill experience from
     it, and come up with a new plan of action.
- Your response should first reason about the search terms, navigation
    decisions, and candidate-selection strategy used in the past
    attempt.
- Then devise a concise, new plan of action that accounts for your
    mistake with reference to specific queries, filters, page
    transitions, or product choices that should have been made.
- Finally, end the response with the distilled experience and improved
    plan inside <remark> </remark> tags.
```

D.4 SCIENCEWORLD

ScienceWorld uses a task-description-conditioned prompt together with a potentially long action-observation history and a dynamic list of currently available actions. In the main setting, the play prompt is additionally constrained by a fixed history-length budget during execution, while the experience distillation prompt is conditioned on the full current-attempt trajectory and the attempt score.

ScienceWorld Execution Prompt

```
You are an expert autonomous agent operating in the ScienceWorld
    environment, which is a text-based virtual environment centered
    around accomplishing tasks from the elementary science curriculum.
Your task is to: {task_description}.
{init_observation}
{experience}
{current_trajectory}

Your admissible actions of the current situation are:
[
{available_actions}
```

```
].

Now it's your turn to take one action for the current step.
Your response should first be step-by-step reasoning about the current
    situation, then think carefully which admissible action best
    advances the task goal.
Once you've finished your reasoning, you should choose an admissible
    action for current step and present it within <action> </action>
    tags.
```

In the experience-distillation prompt, {score} is the integer-valued task-completion score most recently reported by the environment for the current attempt when experience is distilled. This prompt input is distinct from the attempt-level statistic used for VAS, which takes the highest score reached during the attempt, as defined in Appendix B.1.

ScienceWorld Experience Distillation Prompt

```
You are an expert autonomous agent operating in the ScienceWorld
    environment, which is a text-based virtual environment centered
    around accomplishing tasks from the elementary science curriculum.
Your task is to: {task_description}.

You will be given the history of a past attempt.
Your job is to analyze the past sequence, identify any mistakes or
    inefficiencies, and then distill a concise experience with an
    improved plan starting from the original initial state.
{current_trajectory}
Your score was {score}/100 (higher is better). The task is NOT
    successfully completed.

Now it's your turn to analyze the past attempt, distill experience from
     it, and come up with a new plan of action.
- Your response should first reason about the strategy and path you
    took to attempt to complete the task.
- Then devise a concise, new plan of action that accounts for your
    mistake with reference to specific actions that you should have
    taken.
- Finally, end the response with the distilled experience and improved
    plan inside <remark> </remark> tags.
```

D.5 ALFWORLD

ALFWorld uses text observations together with an admissible-action list returned by the environment at each step. In the main same-task evaluation setting, it adopts the execution and experience-distillation prompts shown below. The reported cross-task result uses the sequence-generalization configuration described after these prompts: within each task-type sequence, the agent processes related but distinct task instances sequentially and transfers the distilled experience from the immediately preceding task to the next task together with its fresh initial observation.

ALFWorld Execution Prompt

```
You are an expert agent operating in the ALFRED Embodied Environment.
{init_observation}
{experience}
{current_trajectory}

Your admissible actions of the current situation are:
[{admissible_actions}]
```

```
Now it's your turn to take an action.
- Your response should first reason step-by-step about the current
    situation.
- Once you've finished your reasoning, you should choose an admissible
    action for current step and present it within <action> </action>
    tags.
```

ALFWorld Experience Distillation Prompt

```
You are an expert agent operating in the ALFRED Embodied Environment.
{init_observation}

You will be given the history of a past attempt.
Your job is to analyze the past sequence, identify any mistakes or
    inefficiencies, and then distill a concise experience with an
    improved plan starting from the original initial state.

Below are the actions you took and the corresponding observations:
{current_trajectory}
The task is NOT successfully completed.

Now it's your turn to analyze the past attempt, distill experience from
     it, and come up with a new plan of action.
- Your response should first reason about the strategy and path you
    took.
- Then devise a concise, new plan of action that accounts for your
    mistake with reference to specific actions that you should have
    taken.
- Finally, end the response with the distilled experience and improved
    plan inside <remark> </remark> tags.
```

D.5.1 Cross-Task Experience Transfer

In the reported cross-task evaluation, each sequence contains related but distinct ALFWorld task instances of the same task type. After completing a task, the policy distills transferable experience that captures general strategies for that task type. The next task receives only the distilled experience from the immediately preceding task, together with its fresh initial observation. The current task description, observations, and admissible actions always take priority over the transferred experience.

ALFWorld Cross-Task Experience-Distillation Prompt

```
You are an expert agent operating in the ALFWorld environment.

This is the trajectory from one {task_type} task. The task outcome was
    {outcome}.
Your goal is to distill concise, transferable experience for a
    different task of the
same type.

Below are the actions you took and the corresponding observations:
{current_trajectory}

Please identify the strategies that were effective, the mistakes or
    inefficiencies
that should be avoided, and the general procedure that can be reused
    for another task
of the same type.

Do not include object ids, room details, or answers that are unique to
    the current task
```

```
instance. End with the distilled experience inside <remark> </remark>
    tags.
```

ALFWorld Cross-Task Experience Carrier

```
You previously completed a different task of the same type, and its
    outcome was
{outcome}. The distilled experience from that task is:

{experience}

Use this experience as guidance for the current task, but adapt it to
    the current
situation. The current task description, observations, and admissible
    actions always
take priority.
```

The execution prompt is unchanged between the same-task and cross-task settings. The difference lies in the source of the transferred experience: same-task evaluation transfers experience across attempts on the same task instance, whereas cross-task evaluation transfers the generalizable experience distilled from the immediately preceding task of the same task type.

E CASE STUDY

To provide a more intuitive illustration of how VEL leverages experience from previous failures to improve subsequent decisions under the repeated-attempt setting, we present a representative interaction example from Sokoban in Figure 7. Following the environment setting, each attempt starts from the same initial board state and executes a sequence of decision rounds, where the agent outputs three atomic actions at each round. For clarity, the visualization only presents the terminal board state after each attempt, while omitting intermediate action executions.

The task is successfully completed on the third attempt. In the first attempt, the agent greedily pushes one box onto a nearby target, which appears to make immediate progress but occupies a shared maneuvering corridor required by the remaining box. As a result, the agent reaches an unsolvable state where the second box can no longer be delivered to its target. After this failed attempt, the agent distills an experience from the failed trajectory, identifying the importance of preserving shared maneuvering spaces and considering dependencies between boxes.

In the second attempt, the agent is conditioned on the distilled experience from the previous attempt and avoids blocking the shared corridor. However, it pushes the remaining box against the bottom wall without considering future accessibility. This irreversible push eliminates the required pushing position and leads to another failure. The subsequently distilled experience further identifies the necessity of verifying future pushability before executing irreversible actions.

In the third attempt, the agent is conditioned on the distilled experience from the second attempt and adopts a dependency-aware planning strategy. It preserves critical maneuvering spaces, evaluates the consequences of irreversible pushes, and successfully places both boxes onto their corresponding targets. The trajectory visualization presents the terminal board states of the original task and the three attempts, together with the experience-distillation transitions following the failed attempts. This example demonstrates how VEL transforms concrete failures into abstract planning principles and transfers distilled experience across consecutive attempts. We then provide the experiences distilled after the first two failed attempts.

Corresponding experiences. We reproduce below the experiences distilled from the first and second failed attempts, respectively.

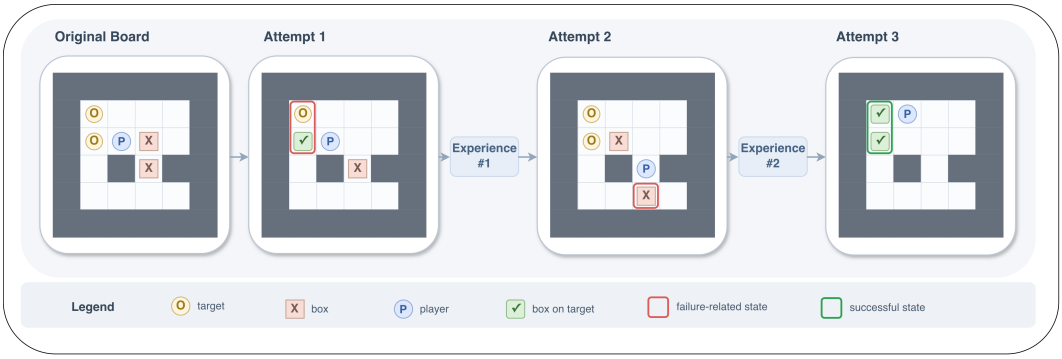

Figure 7: A representative Sokoban example showing how the agent improves across three attempts. The figure presents the terminal board states after each attempt and the corresponding experiences distilled from previous failures.

Experience #1

The mistake in the past experience was committing a box to a target before considering the global dependencies between boxes, which blocked the shared maneuvering corridor and made the remaining box unsolvable. To avoid this, the agent should not focus only on immediate progress. Instead, the agent should preserve shared maneuvering spaces and consider the order of box movements before making irreversible pushes. The agent should first analyze whether placing one box affects the accessibility required for solving other boxes.

Experience #2

The mistake in the past experience was performing an irreversible push without checking whether the box would remain movable afterwards, which pushed the box against the bottom wall and eliminated the required pushing position. To avoid this, the agent should take future consequences into consideration before executing a push. The agent should verify future pushability and preserve the necessary player positions for subsequent box movements. A successful plan should not only achieve immediate progress but also maintain future action flexibility.