Divide and Conquer: Grounding LLMs as Efficient Decision-Making Agents via Offline Hierarchical Reinforcement Learning

Zican Hu [1 2 *] **Wei Liu** [3] **Xiaoye Qu** [2] **Xiangyu Yue** [4] **Chunlin Chen** [1] **Zhi Wang** [1 2 ✉] **Yu Cheng** [4 ✉]

Abstract

While showing sophisticated reasoning abilities, large language models (LLMs) still struggle with long-horizon decision-making tasks due to deficient exploration and long-term credit assignment, especially in sparse-reward scenarios. Inspired by the divide-and-conquer principle, we propose an innovative framework **GLIDER** (**G**rounding **L**anguage Models as Eff**I**cient **D**ecision-Making Agents via Offline Hi**E**rarchical **R**einforcement Learning) that introduces a parameter-efficient and generally applicable hierarchy to LLM policies. We develop a scheme where the low-level controller is supervised with abstract, step-by-step plans that are learned and instructed by the high-level policy. This design decomposes complicated problems into a series of coherent chain-of-thought reasoning sub-tasks, providing flexible temporal abstraction to significantly enhance exploration and learning for long-horizon tasks. Furthermore, GLIDER facilitates fast online adaptation to non-stationary environments owing to the strong transferability of its task-agnostic low-level skills. Experiments on ScienceWorld and ALFWorld benchmarks show that GLIDER achieves consistent performance gains, along with enhanced generalization capabilities.

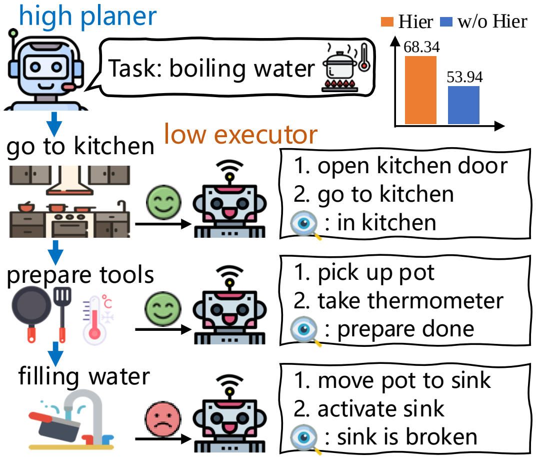

Figure 1: GLIDER's hierarchical framework, showing significant performance gain over non-hierarchical approaches.

1. Introduction

A longstanding goal of artificial general intelligence is to build agents capable of reasoning, decision-making, and communication (Wooldridge & Jennings, 1995; Xu et al., 2024). Recent attempts to exploit large language models (LLMs) as agents have shown commendable results in tackling interactive decision-making tasks (Li et al., 2022; Yao et al., 2023b; Song et al., 2024). Prompt-based methods, like ReAct (Yao et al., 2023b) and Reflexion (Shinn et al., 2023), recursively augment the prompt to a frozen LLM with verbal feedback. They are prone to exceed the input length limit of in-context learning, especially for long-horizon tasks. Scaling with supervised fine-tuning techniques can further unlock the potential of LLMs for downstream applications (Zeng et al., 2023; Chen et al., 2023; Lin et al., 2023). However, their performance is highly dependent on expensive expert demonstrations and can be limited due to deficient exploration of target environments.

Intelligent agents must excel at both imitating demonstrations and adapting behaviors through trial-and-error (Silver et al., 2021; Rafailov et al., 2023). Modern approaches adopt reinforcement learning (RL) algorithms to steer LLMs toward user-specified tasks (Ouyang et al., 2022), such as offline Q-learning (Snell et al., 2023), PPO (Zhai et al., 2024; Szot et al., 2024), or DPO (Song et al., 2024). This paradigm enables SOTA performance in advanced LLMs like OpenAI o1 and DeepSeek-R1 (Guo et al., 2025). However, RL intrinsically requires tedious and vast environment interactions, leading to brittle performance and poor sam-

---

[1]Nanjing University [2]Shanghai AI Laboratory [3]The Hong Kong University of Science and Technology [4]The Chinese University of Hong Kong. * Interns at Shanghai AI Laboratory. ✉ Corresponding Authors: Zhi Wang <zhiwang@nju.edu.cn>, Yu Cheng <chengyu@cse.cuhk.edu.hk>. The code is available at https://github.com/NJU-RL/GLIDER.

Proceedings of the 42st International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

ple efficiency (Burda et al., 2019; Mahankali et al., 2024). Building efficient LLM agents with open-ended textual commands poses several challenges, such as tackling huge action spaces, executing long-horizon planning, and learning from sparse-reward feedback (Rocamonde et al., 2024; Dwaracherla et al., 2024). Existing works still struggle with complex tasks that demand a broad spectrum of vital capabilities, including long-term credit assignment, understanding the real physical world, and sophisticated exploration with structured reasoning (Qiao et al., 2024; Zhou et al., 2024).

Humans naturally tackle complex problems through hierarchical decomposition (Sutton et al., 1999). This divide-and-conquer principle is evident across scales in natural systems, from how corporations divide into specialized departments to how biological systems organize cells to form tissues and organs. The hierarchy design plays a crucial role in advancing frontier research on language agents (Li et al., 2024; Zhou et al., 2024) and embodied intelligence (Ahn et al., 2022; Black et al., 2024), showcasing remarkable efficiency for solving intricate tasks in a more human-like manner.

Inspired by this, we propose an innovative framework **GLIDER** (**G**rounding **L**anguage Models as Eff**I**cient **D**ecision-Making Agents via Offline Hi**E**rarchical **R**L) that introduces a parameter-efficient and generally applicable hierarchy to train competent LLM policies for complex interactive tasks. Our scheme contains two LLM policies, where the low-level controller is supervised to achieve abstract, step-by-step plans learned and proposed by the high-level instructor. By harnessing the strong reasoning and planning capabilities of LLMs, we can decompose complicated problems into a series of coherent chain-of-thought (CoT) reasoning sub-tasks and perform efficient exploration in a semantically structured space. To enhance learning stability, we first build a base agent via behavior cloning, followed by reinforcement fine-tuning of the hierarchical token-level actors and sentence-level critics. This two-level agent is trained in offline mode using hierarchical datasets to achieve prominent sample efficiency, and also can be seamlessly deployed for offline-to-online fine-tuning scenarios.

In summary, our main contributions are as follows:

- Inspired by the divide-and-conquer principle, we propose an offline hierarchical framework GLIDER , empowering LLM agents to tackle complex decision-making tasks via sophisticated exploration and structured reasoning.
- Our method enables fast offline-to-online adaptation to non-stationary environments by developing highly generalizable skills through hierarchical LLM agents.
- Comprehensive studies on ScienceWorld and ALFWorld benchmarks show that our method consistently improves performance and generalization capacity, surpassing a range of baselines by a significant margin.

2. Related Work

LLMs as Decision-Making Agents. With a wealth of semantic knowledge about the world, LLMs have shown remarkable potential in building competent agents across diverse domains (Xi et al., 2023), including reasoning (Wei et al., 2022; Zhou et al., 2023; Luo et al., 2024), robotics (Ahn et al., 2022; Shah et al., 2023) and multi-agent (Chen et al., 2024; Ma et al., 2024). Early studies use a prompt-based framework, such as classical CoT methods (Wei et al., 2022; Yao et al., 2023a; Wang et al., 2023a) that prompt LLMs with intermediate reasoning steps. Follow-up research employs synergy between reasoning and acting (Yao et al., 2023b), incorporates self-reflective verbal feedback (Shinn et al., 2023), and uses strategic reasoning (Gandhi et al., 2023). They use recursive feedback traces to augment prompts, which helps address long sequence and complex, long-horizon task challenges.

Scaling with supervised fine-tuning techniques can unlock the potential for downstream tasks (Hu et al., 2022), such as fine-tuning LLM agents on oracle action trajectories (Lin et al., 2023), on a lightweight instruction-tuning dataset containing high-quality interaction trajectories (Zeng et al., 2023), and on agent trajectories generated from multiple tasks and prompting methods (Chen et al., 2023). Due to inherent limitations, the performance relies heavily on expensive high-quality data and could easily be restricted by deficient exploration of target environments.

RL offers a natural paradigm to unleash the LLM agents' decision-making capabilities (Ouyang et al., 2022). Snell et al. (2023) guides language generation towards maximizing user-specified utility functions using implicit language Q-learning. ETO (Song et al., 2024) collects contrastive trajectory pairs from interactions to update the LLM policy using DPO (Rafailov et al., 2023). Other works employ a similar pipeline where an LLM policy interacts with the environment to receive goal-directed task rewards, which are then used to fine-tune the policy with classical algorithms like PPO (Zhai et al., 2024; Szot et al., 2024; Tan et al., 2024). Building LLM agents with open-ended textual commands can involve a huge action space, and long-horizon planning or sparse-reward scenarios (Mahankali et al., 2024; Zhou et al., 2024). This motivates us to introduce a hierarchy to ground LLMs as efficient decision-making agents.

Hierarchical RL emerges as a powerful framework for managing complexity in decision-making tasks. Classical approaches of Options (Sutton et al., 1999; Bacon et al., 2017) and MAX-Q (Dietterich, 2000) formalize temporal abstractions in RL. Recent advances have significantly expanded these foundations, such as HiRO (Nachum et al., 2018) with data-efficient off-policy training for hierarchical policies, HAC (Levy et al., 2019) with parallel training of 3-level hierarchies, and HiPPO (Li et al., 2020) with efficient

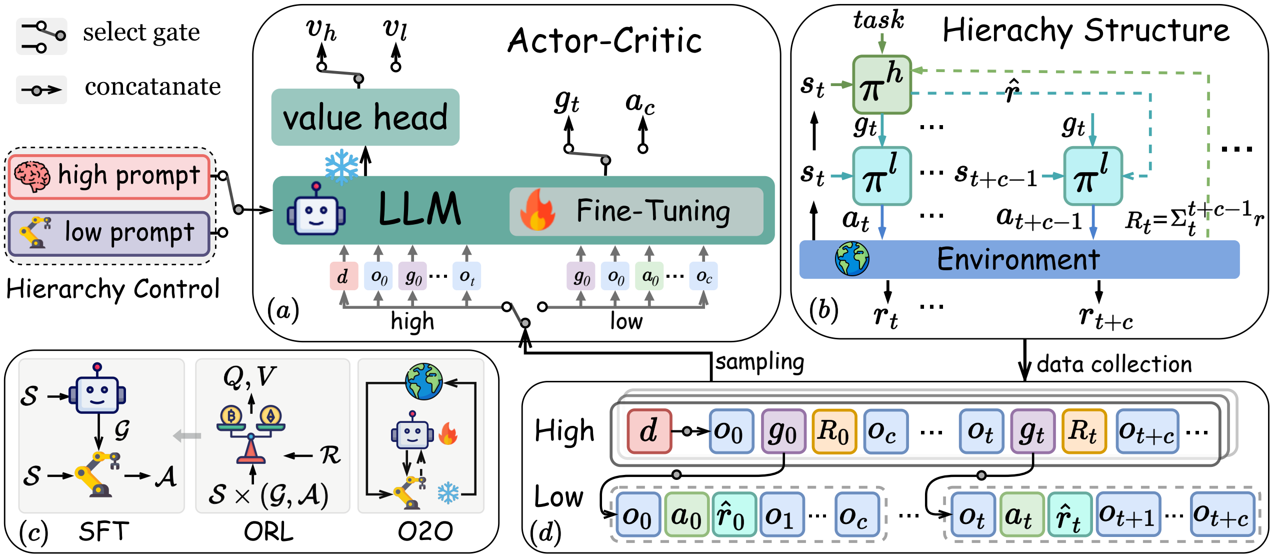

Figure 2: Overview of the GLIDER framework. (a) Hierarchical Actor-Critic architecture with prompt-controlled high- and low-level training on sampled trajectories from offline datasets. (b) Hierarchical policy structure where the high-level π^h generates sub-task g only when the low-level π^l executes primitive actions for c steps. The high-level policy provides the low-level with an intrinsic reward $\hat{r}$ that indicates the sub-task completion, and collects environment rewards across c timesteps as its one-time reward as $R_t = \Sigma r_{t:t+c-1}$. (c) The training pipeline comprises SFT, ORL (offline RL), and O2O (offline-to-online RL) stages. (d) Structured hierarchical trajectories composed of high-level transitions $(d; o_t, g_t, R_t, o_{t+c})$ and low-level transitions $(g; o_t, a_t, \hat{r}_t, o_{t+1})$.

hierarchical policy gradient approximation for robust skill training. In general, a persistent challenge is the dependence on domain expertise to specify meaningful hierarchies. This limitation also motivates us to harness the strong semantic understanding and reasoning abilities of LLMs for natural task decomposition with an autonomous hierarchy.

Offline-to-Online RL. Offline RL (Levine et al., 2020) harnesses offline data without environment interactions, yielding effective algorithms such as BCQ (Fujimoto et al., 2019), CQL (Kumar et al., 2020), and Fisher-BRC (Kostrikov et al., 2021). Also, it remains beneficial to fine-tune the pretrained offline policy with further online interactions, that is, offline-to-online RL (Nair et al., 2020). However, such a benefit is often diminished due to remarkable distribution shifts between pretraining and deployment, leading to accumulated bootstrap errors (Lee et al., 2022; Wang et al., 2023b). Moreover, existing works often require extensive retraining for new tasks (Yu & Zhang, 2023). In contrast, we show that our method achieves fast online adaptation to non-stationary environments via training highly generalizable low-level skills that are robust to distribution shifts.

3. Method

In this section, we present GLIDER, which introduces a parameter-efficient and generally applicable hierarchy to ground LLMs as efficient decision-making agents for tackling complex interactive tasks. Figure 2 illustrates the framework, containing a three-stage pipeline of base agent construction via supervised fine-tuning, policy refinement via offline RL, and seamless adaptation to online deployment. The algorithm pseudocode is given in Appendix A, and detailed implementations are presented as follows.

3.1. Problem Setup of the LLM Agent

We formulate the agent task as a standard Markov decision process (MDP) with a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}/\mathcal{A}$ is the state/action space, $\mathcal{T}/\mathcal{R}$ is the state transition/reward function, and $\gamma \in (0, 1]$ is the discount factor. We introduce an LLM-based policy, $\pi_\theta : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, a probability distribution that maps states to actions, where both $\mathcal{S}$ and $\mathcal{A}$ are drawn from text spaces constrained by user-specified tasks, and policy parameters θ are initialized from a pretrained LLM. The objective is to optimize the policy to receive a maximal expected return as $J(\pi) = \mathbb{E}_\pi[\sum_t \gamma^t r_t]$.

3.2. Hierarchical Architecture of the LLM Agent

We extend the LLM agent setup to a hierarchical two-layer structure, with a high-level policy π_θ^h and a low-level controller π_θ^l. The high-level policy operates at a coarser layer of task planning and sets sub-task goals for the low-level to accomplish. At each timestep t, the environment provides an observation o_t. The high-level planner π_θ^h receives the observation o_t together with a textual task description d, and produces a high-level goal (or sub-task) as $g_t \sim \pi_\theta^h(\cdot \mid d, o_t)$ when $t \equiv 0 \pmod{c}$. This provides temporal abstraction, since high-level decisions are made only every c steps. [1] For the next c timesteps, the low-level controller receives the environment observation o_t and goal g_t, and produces a low-level primitive action as $a_t \sim \pi_\theta^l(\cdot \mid g_t, o_t)$. The action a_t is applied to the environment, which yields a reward r_t and transitions to a new observation o_{t+1}. The high-level policy collects environment rewards through these c timesteps, storing the high-level transition $(d;\ o_t, g_t, \Sigma r_{t:t+c-1}, o_{t+c})$ for offline training. Correspondingly, the high-level dataset for offline learning stages is constructed as

$$\begin{aligned}\mathcal{D}^h = \Sigma_N \left[\, d;\ (o_0, g_0, \Sigma r_{0:c-1}, o_c), ..., \right. \\ \left. (o_t, g_t, \Sigma r_{t:t+c-1}, o_{t+c}), ... \,\right], \end{aligned} \tag{1}$$

which captures strategic task planning over N trajectories.

The high-level policy provides the low-level with an intrinsic reward $\hat{r}$ that indicates the sub-task completion (1 when the sub-task is completed and 0 otherwise).[2] The sub-task completion can be easily accessible from the environment observation o_t, without requiring any manual design or domain knowledge.

Each generated goal g_t corresponds to a sequence of c atomic transitions in the low-level dataset as

$$\begin{aligned}\mathcal{D}^l = \Sigma_N \Sigma_t \left[\, g_t;\ (o_t, a_t, \hat{r}_t, o_{t+1}), ..., \right. \\ \left. (o_{t+c-1}, a_{t+c-1}, \hat{r}_{t+c-1}, o_{t+c}) \,\right], \end{aligned} \tag{2}$$

We design a parameter-efficient hierarchical model architecture. As shown in Figure 2-(a), our model offers superior parameter efficiency from two perspectives. First, the actor and critic share the same frozen LLM backbone, each introducing a minimal number of parameters for efficient fine-tuning at a lightweight computing cost. The actor is formed by augmenting the backbone with LoRA (Hu et al., 2022), which adds a trainable low-rank bypass to each transformer block. The critic is constructed by adding additional MLP layers to the last transformer block of the backbone. Second, the high- and low-level policies share the same actor-critic models, differing only in a *hierarchy prompt* that specifies the level of current inputs. This design benefits from harnessing the powerful capability of LLMs to perform in-context learning, i.e., tackling a series of complex tasks by feeding short prompts to a single foundation model (Brown et al., 2020). In contrast, traditional hierarchical methods usually train independent models at each level, resulting in a multiplication of model parameters (Nachum et al., 2018; Levy et al., 2019; Li et al., 2020).

Our hierarchy setup achieves broad applicability. The hierarchical structure provides temporal abstraction with efficient exploration since high-level decisions are made only when the low-level controller executes for several steps. The high-level policy unlocks the chain-of-thought reasoning of LLMs to decompose a complicated task into a series of coherent sub-task plans, while the low-level model translates abstract plans into precise, executable atomic actions. Our setup achieves generality by training the low-level policy to accomplish sub-task goals learned and instructed by the high-level planner. The high-level planner is guided by environment-provided rewards, while the low-level policy is instructed by the sub-task completion signal derived from environment observations. The whole learning process eliminates the necessity for any manual or task-specific design, making it broadly applicable.

3.3. Base Agent Construction via Behavior Cloning

Directly deploying LLMs as decision-making agents in downstream tasks may generate hallucinatory or inconsistent actions and perform brainless trial-and-error due to the semantic discrepancy between natural language (trained with next-token prediction) and user-specified environments. To improve learning stability and sample efficiency, we first construct a base LLM agent through supervised fine-tuning of the hierarchical actors using pre-collected demonstration trajectories. This behavior cloning process aligns the initial policy with valid action sequences and serves as a solid starting point for building a powerful agent.

Specifically, we imitate the behavior patterns, i.e., the state-to-action mapping function, within both levels of datasets in Eqs. (1) and (2). The objective function of behavior cloning via supervised fine-tuning is to maximize the log-likelihood of the observed data. Pre-trained LLMs tend to generate lengthy, verbose sentences that could be redundant and challenging to understand in the user-specified environment. Hence, we incorporate a length regularization term to encourage the LLM policy to generate concise task plans and atomic actions for effective interaction with the environment. The final loss function is formulated as

$$\begin{aligned}\mathcal{L}_{\text{SFT}}(\theta) = &- \mathbb{E}_{(d,o;g)\sim\mathcal{D}^h}\left[\log \pi_\theta^h(g|d,o)\right] + \lambda \cdot n_h \\ &- \mathbb{E}_{(g,o;a)\sim\mathcal{D}^l}\left[\log \pi_\theta^l(a|g,o)\right] + \lambda \cdot n_l,\end{aligned} \tag{3}$$

where λ is the length regularization ratio, and n_h/n_l is the output length of the high-/low-level policy.

By employing behavior cloning for a base LLM agent construction, we establish a robust foundation for generating

---

[1] c could differ across sub-tasks, as harder sub-tasks naturally require more primitive actions to accomplish.

[2] In a boiling water task, the high-level policy decomposes it into atomic subtasks (e.g., navigation, tools preparation). For navigation subtask, the low-level policy receives a reward of 1 upon reaching kitchen (verified through observation), 0 otherwise.

valid actions with significantly improved sample efficiency. However, the base agent is easily limited by the data quality of demonstration trajectories and lacks the ability to explore the environment. It could easily result in sub-optimal policies, especially when tackling complex long-horizon decision-making challenges.

3.4. Offline Hierarchical Policy Refinement

To unlock the capacity of LLM agents in long-horizon decision-making, we continue to train hierarchical actor-critic models in an offline mode using the reward-annotated datasets $\mathcal{D}^h$ and $\mathcal{D}^l$. The actor outputs a sequence of tokens autoregressively for fine-grained action generation and control at the *token level*, while the critic aims to evaluate the output policy at the *sentence level*. In the following, we use s and u to denote the state and action uniformly, i.e., $s = (d, o), u = g$ for the high-level policy and $s = (g, o), u = a$ for the low-level.

Sentence-Level Critic. Following the practice in advanced offline RL algorithms (Snell et al., 2023; Zhou et al., 2024), the critic component consists of a Q-function $Q_\phi(s, u)$ and a value function $V_\psi(s)$ that are optimized using temporal difference learning. The Q-function is trained to minimize the Bellman bootstrapping error as

$$\mathcal{L}_Q(\phi)=\mathbb{E}_{(s,u,r,s')\sim D_r}\Big[\big(r+\gamma V_{\bar{\psi}}(s')-Q_\phi(s,u)\big)^2\Big], \quad (4)$$

where the Bellman target is computed from a delayed copy of the value model $V_{\bar{\psi}}$. The value function is trained using an asymmetric loss function to maintain a conservative value estimation as

$$\mathcal{L}_V(\psi)=\mathbb{E}_{s\sim D_r}\big[\mathbb{E}_{u\sim\pi_\theta(\cdot|s)}\left[L_2^\tau\big(Q_{\bar{\phi}}(s,u)-V_\psi(s)\big)\right]\big], \quad (5)$$

where $L_2^\tau(x) = |\tau - \mathbb{1}(x < 0)|x^2$ is an asymmetric loss function with a expectile parameter $\tau \in [0.5, 1)$, introduced in implicit Q-learning (Kostrikov et al., 2022). This loss function assigns more importance to $Q > V$ predictions (weighted by τ) while reducing the influence of $Q < V$ predictions (weighted by $1-\tau$). The asymmetric design helps prevent the learned value function from being overly optimistic, as overestimation could easily lead to poor policy updates in offline RL settings due to distribution shift. The delayed target networks $\bar{\psi}$ and $\bar{\phi}$ are periodically updated using Polyak averaging (Haarnoja et al., 2018) to improve training stability.

Token-Level Actor. The LLM-based actor outputs a sequence of tokens $w_{1:n}$ autoregressively, where n is the output sentence length. Each token is selected according to the token probability distribution generated by token actor as

$$\pi_\theta(u \mid s) = \pi_\theta(w_{1:n} \mid s) = \prod_{i=1}^{n} \pi_\theta(w_i \mid s, w_{1:i-1}). \quad (6)$$

The actor is trained to maximize the expected return of the policy, i.e., the estimated Q-function, which is also equivalent to maximizing the advantage function. Following the practice in AWAC (Nair et al., 2020), we formulate the policy optimization of the token actor as a weighted maximum likelihood estimation problem. The resulting loss function is derived as

$$\begin{aligned}\mathcal{L}_\pi(\theta) &= -\mathbb{E}_{(s,u)\sim D_r}\left[\exp\left(\frac{1}{\lambda}A(s,u)\right)\cdot\log\pi_\theta(u\mid s)\right]\\ &= -\mathbb{E}_{(s,u)\sim D_r}\Bigg[\exp\left(\frac{1}{\lambda}\left(Q_\phi(s,u)-V_\psi(s)\right)\right) \quad (7)\\ &\qquad\cdot\sum_{i=1}^{n}\log\pi_\theta(w_i\mid s,w_{1:i-1})\Bigg].\end{aligned}$$

This "supervised" formulation implicitly enforces a constraint to mitigate distribution shift and avoids overly conservative updates with advantage weighting, thus facilitating efficient hierarchical policy learning from offline data. Further, by eliminating over-conservatism and explicit modeling of the behavior policy, it is well suited to perform fast adaptation to new tasks in online deployment, as studied in Sec. 3.5.

3.5. Offline-to-Online Adaptation

In the offline stage, we decompose the LLM agent into a series of low-level sub-tasks (or skills) and a high-level policy with strong reasoning abilities. With this flexible hierarchical structure, GLIDER can be efficiently adapted to new environments with further online interactions in offline-to-online scenarios. The low-level skills are pre-trained using intrinsic reward functions rather than task-specific ones, allowing for high generalization capacity across tasks and good robustness to the distribution shift between offline pre-training and online deployment. Naturally, we freeze the task-agnostic low-level skills that interact with the new environment, and only fine-tune the high-level policy with the environment-provided reward signals.

Formally, at each timestep t, the high-level policy receives the environment observation o_t together with the task description d, and selects a low-level skill as $g_t \sim \pi_\theta^h(\cdot|d, o_t)$. Then, the fixed skill g_t interacts with the environment for c primitive actions, resulting in a c-step trajectory as $[g_t;\quad (o_t, a_t, r_t, o_{t+1}), ..., (o_{t+c-1}, a_{t+c-1}, r_{t+c-1}, o_{t+c})]$. We construct the transition sample for the high-level policy as $(d;\ o_t, g_t, \Sigma r_{t:t+c-1}, o_{t+c})$. Finally, we collect these transition samples to fine-tune the high-level critic with Eqs. (4)-(5) and the actor with Eq. (7). By harnessing the temporal abstraction knowledge embodied in the pretrained low-level skills, GLIDER can quickly adapt to non-stationary environments with significantly improved exploration efficiency.

4. Experiments

We evaluate GLIDER in offline settings from Sec. 4.2 to Sec. 4.5, and then test its adaptability in an online fine-tuning manner in Sec. 4.4. Through comprehensive experiments, we aim to answer the following research questions:

- How effective and robust is GLIDER across diverse settings? We examine its performance against prompt-based and fine-tuning baselines, assess consistency across different backbones, and evaluate agent capacity in both sparse and dense reward environments. (See Sec. 4.2).
- What is the contribution of each component to GLIDER's performance? Through systematic ablation studies, we analyze the impact of the hierarchical structure, training stages (SFT and ORL), and variations in model architecture and scale. (See Sec. 4.3).
- How well does GLIDER generalize to out-of-domain tasks through online fine-tuning? We evaluate the model's adaptation capabilities on previously unseen task distributions. (See Sec. 4.4).
- How do varying ratios of expert demonstrations to medium-quality data affect model performance? We evaluate different mixture strategies for the composition of training data. (See Sec. 4.5).

4.1. Experimental Settings

Benchmarks and Offline Dataset. We evaluate GLIDER on two popular language-based interactive decision-making tasks: 1) `ScienceWorld` (Wang et al., 2022) is a textual environment for elementary science experiments, featuring 30 tasks. 2) `ALFWorld` (Shridhar et al., 2021) contains 6 types of household manipulation tasks, requiring agents to navigate and interact with objects following language instructions in a binary reward setting.

For offline training data, we construct a dataset that combines expert demonstrations (optimal trajectories provided by benchmarks) and medium-quality trajectories with a mixture ratio of $1:2$. The medium-quality trajectories are collected through two strategies: in-distribution and cross-task generalization sampling. Appendix B presents more details of benchmarks and offline dataset construction.

Models and Baselines. We build our method on three open-source language models: 1) `Mistral-7B` (Jiang et al., 2023), 2) `Gemma-7B` (Team et al., 2024) and 3) `Llama-3-8B` (Meta, 2024). We employ LoRA for parameter-efficient fine-tuning of all models. Appendix C presents detailed model architectures, hyperparameters, and training and evaluation setups.

We compare GLIDER against various strong baselines: 1) `ReAct` (Yao et al., 2023b), a pioneering approach that incorporates CoT prompting in decision-making tasks through a structured Thought-Action-Observation loop. 2) `Reflexion` (Shinn et al., 2023), an advanced prompt-based framework that enhances agent decision-making through self-reflective verbal feedback. 3) `SwiftSage` (Lin et al., 2023), a dual-process cognitive framework that integrates the strengths of behavior cloning and prompting for complex interactive reasoning and action-planning tasks. 4) `NAT` (Wang et al., 2024), a fine-tuning approach that enables LLMs to learn from failure trajectories through data quality control. 5) `ETO` (Song et al., 2024), an iterative optimization framework between exploring the environment to collect contrastive trajectory pairs and fine-tuning the policy using DPO (Rafailov et al., 2023).

4.2. Primary Performance

Table 1 illustrates the comprehensive evaluation results of GLIDER across three backbone models (Mistral-7B, Gemma-7B, and Llama-3-8B) on both ScienceWorld and ALFWorld benchmarks, compared to competent prompt-based methods (ReAct, Reflexion, and SwitchSage) and fine-tuning approaches (NAT and ETO). Generally, fine-tuning approaches yield better results than prompt-based methods, and GLIDER further exceeds these strong baselines by a significant margin in both seen and unseen tasks across diverse settings. Taking ScienceWorld as an example, GLIDER obtains the best performance with Llama-3-8B as the backbone, achieving high scores of 77.43 (+33.73%) on seen tasks and 68.34 (+30.59%) on unseen tasks. Similar improvements are observed with Mistral-7B (+15.71% on seen tasks, +25.63% on unseen tasks) and Gemma-7B (+26.23% on seen tasks, +22.28% on unseen tasks). Notably, the substantial performance gains on unseen tasks (ranging from +22.28% to +30.59%) highlight GLIDER's impressive generalization capability, which is essential for modern AI agents. In summary, the consistent performance improvement across diverse model architectures and benchmarks validates the effectiveness and robustness of our method.

4.3. Ablation Studies

We conduct extensive studies to analyze the respective contributions of the hierarchical structure and learning stages in GLIDER, yielding five ablations: 1) `w/ Hier(SFT)`, it ablates the offline RL stage and trains a hierarchical agent with SFT only; 2) `w/ Hier(ORL)`, it ablates the SFT stage and trains a hierarchical agent with offline RL only; 3) `w/o Hier(SFT+ORL)`, it ablates the hierarchy and trains a single-layer agent with SFT and offline RL; 4) `w/o Hier(SFT)`, it ablates the hierarchy and the offline RL stage, training a single-layer agent with SFT only; 5) `w/o Hier(ORL)`, it ablates the hierarchy and the SFT stage, training a single-layer agent with offline RL only.

Table 1: **Main Results.** Performance comparison across three backbone models on ScienceWorld and AlfWorld benchmarks. ○ indicates prompt-based methods without model parameter update, while ◐ represents fine-tuning approaches using LoRA. ↑ denotes the performance improvement of GLIDER compared to the best results among the baselines.

Backbone	**Method**	**ScienceWorld**		**AlfWorld**	
		Seen	Unseen	Seen	Unseen
Mistral-7B	○ ReAct	20.72	17.65	7.86	5.22
	○ Reflexion	21.07	18.11	11.56	6.00
	○ SwitchSage	48.40	45.25	30.29	26.52
	◐ NAT	57.12	50.79	64.43	68.96
	◐ ETO	58.17	51.85	66.84	71.43
	◐ **GLIDER**	**67.31** (↑ 15.71%)	**65.14** (↑ 25.63%)	**70.02** (↑ 4.76%)	**74.83** (↑ 4.76%)
Gemma-7B	○ ReAct	3.58	3.51	6.43	2.24
	○ Reflexion	4.94	3.93	7.14	2.99
	○ SwitchSage	33.43	30.90	8.23	5.72
	◐ NAT	47.63	44.98	67.86	65.88
	◐ ETO	50.44	47.84	66.43	68.66
	◐ **GLIDER**	**63.67** (↑ 26.23%)	**58.50** (↑ 22.28%)	**72.12** (↑ 6.28%)	**70.88** (↑ 3.23%)
Llama-3-8B	○ ReAct	24.76	22.66	2.86	3.73
	○ Reflexion	27.23	25.41	4.29	4.48
	○ SwitchSage	42.22	40.58	20.39	10.78
	◐ NAT	55.24	48.76	60.71	59.70
	◐ ETO	57.90	52.33	64.29	64.18
	◐ **GLIDER**	**77.43** (↑ 33.73%)	**68.34** (↑ 30.59%)	**71.56** (↑ 11.31%)	**75.38** (↑ 17.45%)

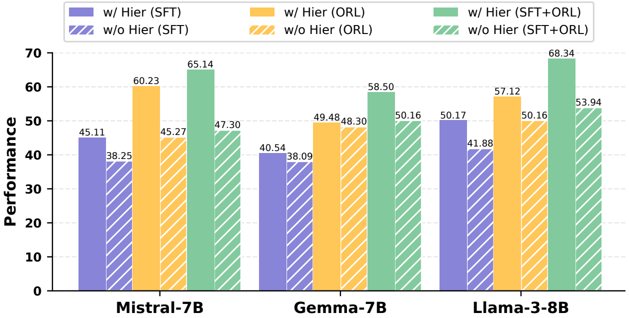

Figure 3: Ablation performance on unseen tasks in ScienceWorld across model architectures. Solid pillars denote hierarchical models and shaded pillars indicate ablating the hierarchy. The purple/yellow/green pillars correspond to SFT/ORL/SFT+ORL training stages, respectively.

Ablation across Model Architectures. Figure 3 presents the performance on unseen tasks in ScienceWorld. We conduct these ablations across different language models to ensure robust findings. First, the hierarchical structure plays a crucial part in all training stages, as models incorporating hierarchy outperform their non-hierarchical counterparts by significant margins. The improvement is most pronounced in the full stage of SFT+ORL (green), followed by the ORL stage (yellow), and finally the SFT setting (purple). This interesting phenomenon highlights the superiority of our method as a whole. Another interesting observation is that training offline RL agents from scratch (yellow) performs better than training SFT agents (purple). It highlights the higher potential of reinforcement fine-tuning over supervised fine-tuning, akin to the observation in DeepSeek-R1 (Guo et al., 2024). Initializing ORL from SFT parameters (green) proves to be a more effective strategy, which is also consistent to the common practice in literature (Silver et al., 2016; Song et al., 2024). Moreover, using different backbones exhibits similar patterns in these ablations, while Mistral-7B and Llama-3-8B induce better performance compared to Gemma-7B. In summary, these results validate the effectiveness of both the hierarchical structure and the multi-stage training in GLIDER, with their combination yielding the most significant results across all implementations.

Table 2: Ablation performance on unseen tasks in ScienceWorld across model scales. [3]

Model	**w/o Hier**			**w/ Hier**		
	SFT	ORL	SFT+ORL	SFT	ORL	SFT+ORL
Llama-1B	37.24	45.31	48.48	44.50	50.43	53.62
Llama-3B	38.19	52.47	56.93	48.11	55.98	61.29
Llama-8B	41.88	50.16	53.94	50.17	57.12	68.34

Ablation across Model Scales. Further, we investigate the impact of model scales on ablation performance. Table 2 presents the ablation results on ScienceWorld's unseen tasks with Llama models ranging from 1B to 8B parameters, consistently demonstrating the advantages of the hierarchical structure and multi-stage training pipeline. No-

---

[3] Llama-1B and Llama-3B models refer to the Meta-Llama-3.2-1B-Instruct and Meta-Llama-3.2-3B-Instruct version, respectively.

tably, our hierarchical approach demonstrates remarkable efficiency even with small parameter counts. Taking the w/ Hier (SFT+ORL) as an example, Llama-3B achieves a score of 61.29, surpassing even larger models like Mistral-7B with a score of 58.50. This suggests that our hierarchical structure effectively enhances agent capacity without necessitating particularly large parameter counts, making our method more practical for resource-constrained scenarios.

4.4. Generalization Analysis via Online Fine-tuning

To evaluate GLIDER's generalization capacity to non-stationary environments, we test it in offline-to-online fine-tuning scenarios with comparison to the traditional AC algorithm (Konda & Tsitsiklis, 1999) and the classical offline-to-online algorithm AWAC (Nair et al., 2020). In the ScienceWorld benchmark, we categorize the science experiment tasks into three distinct domains: electrical, biology, and thermodynamics. From each group, we exclude one representative task (test-conductivity, find-animal, and boil) during offline training, and observe the trained agent's adaptation performance with online fine-tuning on that task. As shown in Figure 4, GLIDER exhibits strong generalization ability to new tasks in at least two aspects. First, GLIDER achieves a higher initial test score, highlighting its superior zero-shot generalization capacity and enhanced knowledge transfer to new online tasks. Second, during the online fine-tuning process, GLIDER shows significantly faster adaptation and achieves substantially better final performance on all tasks. In summary, these results comprehensively validate GLIDER's superior generalization capability to new tasks in both zero-shot knowledge transfer and fast online adaptation, establishing GLIDER as a more competent language agent with impressive autonomous adaptability.

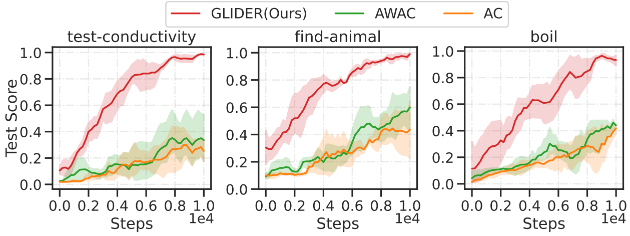

Figure 4: Online fine-tuning performance (score/100) of GLIDER against AC and AWAC baselines in ScienceWorld.

4.5. Impact of Data Mixture Ratios

We investigate how different mixture ratios between expert and medium data affect agent performance during offline RL training. Figure 5 presents the GLIDER's performance (with and without hierarchy) on unseen tasks in ScienceWorld across data mixture ratios. The agent achieves satisfactory capabilities when the expert-to-medium data ratio falls between 2:1 to 1:5, performing the best at 1:2 with a score of 68.3. An interesting phenomenon is that training with only expert demonstrations results in a limited performance of score 29.7, and training solely on medium data can obtain a slightly higher score of 36.0. It suggests that increasing the trial-and-error experience and coverage of the state-action space (expert data is somewhat homogeneous) might facilitate the generalization performance on unseen tasks. Compared to supervised learning, RL naturally learns from sub-optimal data and continually reinforcing its capabilities through self-evolving. This finding also supports our motivation to boost the LLM agent's competence via RL. In summary, both the data quality and diversity are crucial for building capable and generalizable LLM agents.

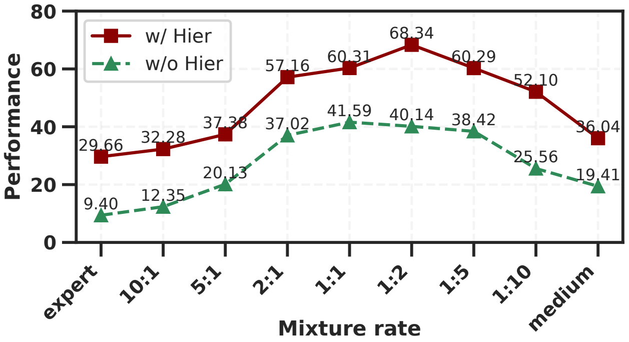

Figure 5: Performance on unseen tasks in ScienceWorld with different expert-to-medium data mixture ratios in the offline RL stage with Llama-3-8B as the LLM backbone.

5. Conclusions, Limitations, and Future Work

We propose GLIDER, an innovative framework that empowers LLM agents with high-capacity decision-making abilities through offline hierarchical RL. We design a concise hierarchical model architecture that achieves superior parameter efficiency and broad applicability, efficiently grounding LLM agents to tackle complex, long-horizon tasks via sophisticated exploration and structured reasoning. Extensive experiments validate GLIDER's consistent improvement on learning performance and generalization capability.

Though, our method employs a multi-stage pipeline that involves a somewhat complex training procedure. A promising future work is to streamline the training pipeline while maintaining high efficiency, inspired by DeepSeek-R1's recent advances in reinforcement fine-tuning. Further, our framework's potential can extend beyond strict agent tasks, since many LLM tasks can also be reformulated as the sequential decision-making paradigm through process reward model (PRM). A crucial future step is to extend our method to broader domains such as mathematical reasoning and code generation tasks, unleashing the inherent capability of hierarchical agents in addressing complicated problems.

Acknowledgements

We thank Xuyang Hu and Zhenhong Sun for their helpful discussions. This work was supported in part by the National Natural Science Foundation of China (Nos. 62376122 and 72394363), in part by the AI & AI for Science Project of Nanjing University, and in part by the Nanjing University Integrated Research Platform of the Ministry of Education-Top Talents Program.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., et al. Do as I can, not as I say: Grounding language in robotic affordances. In *Proceedings of Conference on Robot Learning*, 2022.

Bacon, P.-L., Harb, J., and Precup, D. The option-critic architecture. In *Proceedings of AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Black, K., Brown, N., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Groom, L., Hausman, K., Ichter, B., et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. In *Proceedings of International Conference on Learning Representations*, 2019.

Chen, B., Shu, C., Shareghi, E., Collier, N., Narasimhan, K., and Yao, S. FireAct: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*, 2023.

Chen, W., Su, Y., Zuo, J., Yang, C., Yuan, C., Chan, C.-M., Yu, H., Lu, Y., Hung, Y.-H., Qian, C., et al. AgentVerse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *Proceedings of International Conference on Learning Representations*, 2024.

Dietterich, T. G. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

Dwaracherla, V., Asghari, S. M., Hao, B., and Van Roy, B. Efficient exploration for llms. In *Proceedings of International Conference on Machine Learning*, pp. 12215–12227, 2024.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *Proceedings of International Conference on Machine Learning*, pp. 2052–2062, 2019.

Gandhi, K., Sadigh, D., and Goodman, N. D. Strategic reasoning with language models. *arXiv preprint arXiv:2305.19165*, 2023.

Guo, D., Zhu, Q., Yang, D., Xie, Z., Dong, K., Zhang, W., Chen, G., Bi, X., Wu, Y., Li, Y., et al. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*, 2024.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., et al. DeepSeek-R1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of International Conference on Machine Learning*, pp. 1861–1870, 2018.

Hu, E. J., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., et al. LoRA: Low-rank adaptation of large language models. In *Proceedings of International Conference on Learning Representations*, 2022.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Konda, V. and Tsitsiklis, J. Actor-critic algorithms. In *Advances in Neural Information Processing Systems*, volume 12, pp. 75993–76005, 1999.

Kostrikov, I., Fergus, R., Tompson, J., and Nachum, O. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*, pp. 5774–5783, 2021.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit Q-learning. In *Proceedings of International Conference on Learning Representations*, 2022.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1179–1191, 2020.

Lee, S., Seo, Y., Lee, K., Abbeel, P., and Shin, J. Offline-to-online reinforcement learning via balanced replay and pessimistic Q-ensemble. In *Proceedings of Conference on Robot Learning*, pp. 1702–1712, 2022.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Levy, A., Konidaris, G., Platt, R., and Saenko, K. Learning multi-level hierarchies with hindsight. In *Proceedings of International Conference on Learning Representations*, 2019.

Li, A., Florensa, C., Clavera, I., and Abbeel, P. Sub-policy adaptation for hierarchical reinforcement learning. In *Proceedings of International Conference on Learning Representations*, 2020.

Li, S., Puig, X., Paxton, C., Du, Y., Wang, C., Fan, L., Chen, T., Huang, D.-A., Akyürek, E., Anandkumar, A., et al. Pre-trained language models for interactive decision-making. In *Advances in Neural Information Processing Systems*, volume 35, pp. 31199–31212, 2022.

Li, Z., Xie, Y., Shao, R., Chen, G., Jiang, D., and Nie, L. Optimus-1: Hybrid multimodal memory empowered agents excel in long-horizon tasks. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Lin, B. Y., Fu, Y., Yang, K., Brahman, F., Huang, S., Bhagavatula, C., Ammanabrolu, P., Choi, Y., and Ren, X. SwiftSage: a generative agent with fast and slow thinking for complex interactive tasks. In *Advances in Neural Information Processing Systems*, pp. 23813–23825, 2023.

Luo, Z., Xu, C., Zhao, P., Sun, Q., Geng, X., Hu, W., Tao, C., Ma, J., Lin, Q., and Jiang, D. WizardCoder: Empowering code large language models with evol-instruct. In *Proceedings of International Conference on Learning Representations*, 2024.

Ma, W., Mi, Q., Zeng, Y., Yan, X., Wu, Y., Lin, R., Zhang, H., and Wang, J. Large language models play StarCraft II: Benchmarks and a chain of summarization approach. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Mahankali, S., Hong, Z.-W., Sekhari, A., Rakhlin, A., and Agrawal, P. Random latent exploration for deep reinforcement learning. In *Proceedings of International Conference on Machine Learning*, pp. 34219–34252, 2024.

Meta. Introducing Meta Llama 3: The most capable openly available LLM to date, 2024. `https://ai.meta.com/blog/meta-llama-3/`.

Nachum, O., Gu, S. S., Lee, H., and Levine, S. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 31, pp. 3307–3317, 2018.

Nair, A., Gupta, A., Dalal, M., and Levine, S. AWAC: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022.

Qiao, S., Fang, R., Zhang, N., Zhu, Y., Chen, X., Deng, S., Jiang, Y., Xie, P., Huang, F., and Chen, H. Agent planning with world knowledge model. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741, 2023.

Rocamonde, J., Montesinos, V., Nava, E., Perez, E., and Lindner, D. Vision-language models are zero-shot reward models for reinforcement learning. In *Proceedings of International Conference on Learning Representations*, 2024.

Shah, D., Osiński, B., Levine, S., et al. LM-Nav: Robotic navigation with large pre-trained models of language, vision, and action. In *Proceedings of Conference on Robot Learning*, pp. 492–504, 2023.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K. R., and Yao, S. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023.

Shridhar, M., Yuan, X., Cote, M.-A., Bisk, Y., Trischler, A., and Hausknecht, M. ALFWorld: Aligning text and embodied environments for interactive learning. In *Proceedings of International Conference on Learning Representations*, 2021.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Silver, D., Singh, S., Precup, D., and Sutton, R. S. Reward is enough. *Artificial Intelligence*, 299:103535, 2021.

Snell, C. V., Kostrikov, I., Su, Y., Yang, S., and Levine, S. Offline RL for natural language generation with implicit language Q-learning. In *Proceedings of International Conference on Learning Representations*, 2023.

Song, Y., Yin, D., Yue, X., Huang, J., Li, S., and Lin, B. Y. Trial and error: Exploration-based trajectory optimization for llm agents. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, 2024.

Sutton, R. S., Precup, D., and Singh, S. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211, 1999.

Szot, A., Schwarzer, M., Agrawal, H., Mazoure, B., Metcalf, R., Talbott, W., Mackraz, N., Hjelm, R. D., and Toshev, A. T. Large language models as generalizable policies for embodied tasks. In *Proceedings of International Conference on Learning Representations*, 2024.

Tan, W., Zhang, W., Liu, S., Zheng, L., Wang, X., and An, B. True knowledge comes from practice: Aligning large language models with embodied environments via reinforcement learning. In *Proceedings of International Conference on Learning Representations*, 2024.

Team, G., Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., Sifre, L., Rivière, M., Kale, M. S., Love, J., et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Wang, L., Xu, W., Lan, Y., Hu, Z., Lan, Y., Lee, R. K.-W., and Lim, E.-P. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pp. 9–14, 2023a.

Wang, R., Jansen, P., Côté, M.-A., and Ammanabrolu, P. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of Empirical Methods in Natural Language Processing*, pp. 11279–11298, 2022.

Wang, R., Li, H., Han, X., Zhang, Y., and Baldwin, T. Learning from failure: Integrating negative examples when fine-tuning large language models as agents. *arXiv preprint arXiv:2402.11651*, 2024.

Wang, S., Yang, Q., Gao, J., Lin, M., Chen, H., Wu, L., Jia, N., Song, S., and Huang, G. Train once, get a family: State-adaptive balances for offline-to-online reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 47081–47104, 2023b.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022.

Wooldridge, M. and Jennings, N. R. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.

Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., et al. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023.

Xu, Z., Yu, C., Fang, F., Wang, Y., and Wu, Y. Language agents with reinforcement learning for strategic play in the werewolf game. In *Proceedings of International Conference on Machine Learning*, pp. 55434–55464, 2024.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. R. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822, 2023a.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of International Conference on Learning Representations*, 2023b.

Yu, Z. and Zhang, X. Actor-critic alignment for offline-to-online reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 40452–40474, 2023.

Zeng, A., Liu, M., Lu, R., Wang, B., Liu, X., Dong, Y., and Tang, J. AgentTuning: Enabling generalized agent abilities for LLMs. *arXiv preprint arXiv:2310.12823*, 2023.

Zhai, Y., Bai, H., Lin, Z., Pan, J., Tong, S., Zhou, Y., Suhr, A., Xie, S., LeCun, Y., Ma, Y., et al. Fine-tuning large vision-language models as decision-making agents via reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q. V., and Chi, E. H. Least-to-most prompting enables complex reasoning in large language models. In *Proceedings of International Conference on Learning Representations*, 2023.

Zhou, Y., Zanette, A., Pan, J., Levine, S., and Kumar, A. ArCHer: Training language model agents via hierarchical multi-turn rl. In *Proceedings of International Conference on Machine Learning*, pp. 62178–62209, 2024.

Appendix A. Algorithm Pseudocodes

Based on the implementations in Section 3, we summarize the brief procedure of GLIDER. Algorithm 1 presents the complete training pipeline of GLIDER, which consists of three stages. In SFT stage, we perform behavioral cloning to train both high-level and low-level policies using demonstration data. Notably, both policies share the same LLM parameters but are differentiated through distinct prompts, which significantly reduces the parameter count while maintaining the hierarchical structure. The high-level policy prompt focuses on task decomposition, while the low-level policy prompt emphasizes primitive action generation. In ORL Stage, where both policies and critics are updated using data from high-level ($\mathcal{B}_\mathcal{H}$) and low-level ($\mathcal{B}_\mathcal{L}$) replay buffers, where contains a balanced mixture of expert demonstrations and medium-quality trajectories. Critics are updated through bootstrapping, while policies are optimized via a policy gradient. O2O stage describes the optional online adaptation process. Due to parameter sharing between policies, we cannot strictly fix the low-level policy parameters. Instead, we maintain low-level policy performance by continually training on offline demonstration data while simultaneously fine-tuning the high-level policy using newly collected transition data.

```
Algorithm 1: GLIDER: A Hierarchical Framework for LLM-based Decision Making
Input:
• High-level replay buffer: $\mathcal{B}_\mathcal{H} = \{(d, o_0, g_0, R_0, o_c..., o_t, g_t, R_t, o_{t+c})^{(i)}\}$, where $R_t = \sum_{i=t}^{t+c-1} r_i$
• Low-level replay buffer: $\mathcal{B}_\mathcal{L} = \{(g_t, o_t, a_t, \hat{r}_t, o_{t+1}, ..., o_{t+c-1}, a_{t+c-1}, \hat{r}_{t+c-1}, o_{t+c})^{(j)}\}$
• Environment: env
• Hierarchical policy: $\pi_\theta^h, \pi_\theta^l$
• Hierarchical critic: $Q_\phi^h, V_\psi^h, Q_\phi^l, V_\psi^l$
• Hyperparameters: discount $\gamma$, update rate $\tau$, regularization weight $\lambda$

Output: Optimized hierarchical policy $\pi_\theta = \{\pi_\theta^h, \pi_\theta^l\}$
// Stage 1: SFT
for iteration $i = 1, 2, ...$ do
    Update hierarchical policy via BC loss (Eq.3,Eq.6)
end
// Stage 2: ORL
for iteration $i = 1, 2, ...$ do
    Sample batches from $\mathcal{B}_\mathcal{H}$ and $\mathcal{B}_\mathcal{L}$
    Update critics via bootstrapping (Eq.4, Eq.5)
    Update policies via policy gradient (Eq.7)
    Soft update target networks: $\bar{\eta} \leftarrow (1-\tau)\bar{\eta} + \tau\eta$
end
// Stage 3: O2O (Optional) Fix low-level policy $\pi_\theta^l$
for episode $= 1, 2, ...$ do
    $o_0 \leftarrow$ env.reset(task), $d \leftarrow$ env.task_description()
    Initialize trajectory $\xi \leftarrow \varnothing$
    for $t = 1, ..., T$ do
        Sample subtask: $g_t \sim \pi_\theta^h(\cdot \mid d, o_t)$
        for step $h = 1, ..., c$ do
            Sample action: $a_t \sim \pi_\theta^l(\cdot \mid g_t, o_t)$
            $r_t, o_{t+1}, done \leftarrow$ env.step($a_t$)
        end
        Store transition: $\xi \leftarrow \xi \cup (d, o_t, g_t, R_t, o_t + c)$
        $t \leftarrow t + c$
    end
    Store trajectory: $\mathcal{B}_\mathcal{H} \leftarrow \mathcal{B}_\mathcal{H} \cup \{\xi\}$
    Update high-level policy and critic using $\mathcal{B}_\mathcal{H}$ (Eq.4-7)
end
```

Appendix B. Dataset Information

Benchmarks.

We evaluate GLIDER on two popular language-based interactive decision-making tasks:

1. **ScienceWorld** (Wang et al., 2022) is a textual environment for elementary science experiments, featuring 30 tasks across 10 categories. Agents must demonstrate scientific understanding through interactive experimentation, with progress measured by a dense reward (0 to 1) at each step.
2. **ALFWorld** (Shridhar et al., 2021) simulates household environments that require navigation and object manipulation in a sparse, binary reward setting. The reward is 1 only upon successful task completion, and 0 otherwise.

Beyond standard evaluation on seen tasks, it includes unseen scenarios to assess generalization ability. Table 3 presents the statistical information of our datasets. Both ScienceWorld and ALFWorld contain Text-Seen and Text-Unseen test sets, where Text-Unseen comprises out-of-distribution variations to evaluate the generalization capabilities of different agents.

Table 3: Dataset statistics.

Dataset	Train	Text-Seen	Text-Unseen
ScienceWorld	1,483	194	211
ALFWorld	3,119	140	134

Expert Demonstration

To support imitation learning, the ScienceWorld and ALFWorld provide human-annotated trajectories. For hierarchical data structuring, we utilize GPT-4 to decompose these trajectories into subtasks, creating a clear hierarchical representation of the demonstration data. We show an example expert demonstration trajectory for w/o and w/ hierarchical in ScienceWorld in Figure. 6

Medium Data Collection

For medium-quality data collection, we employ two distinct sampling strategies:

1. **In-distribution Sampling:** During the SFT training process, we utilize the intermediate policy to sample sub-optimal trajectories on the training tasks.This sampling strategy helps better cover the task's world model, as the intermediate policy explores diverse solution paths and state transitions, leading to a more comprehensive understanding of the environment dynamics and task structure.
2. **Cross-task Generalization Sampling:** We partition the training tasks into n (we set $n = 3$) subsets. For each iteration, we train an SFT model on one subset and use it to collect data on the unseen training tasks from other subsets. This sampling strategy is specifically designed to capture trajectories that exhibit better generalization capabilities, as the policy is forced to adapt to novel but related tasks during data collection.

Both of strategies approach ensures diversity in our medium-quality dataset, combining both task-specific expertise and cross-task generalization abilities. We provide examples of both strategies with and without hierarchical structure in Figure 7 and Figure 8, respectively.

In Figure 7 (in-distribution sampling), we can see the the exploration without hierarchy terminates early after attempting to `"pick up apple juice"`. In contrast, the hierarchical approach extends further through clear subtask planning (`"Prepare tools"`, `"Find apple juice"`, `"Heat apple juice"`), continuing until it discovers the stove malfunction.

In Figure 8 (cross-task generalization sampling) demonstrates enhanced exploration capabilities. Under the same initial conditions, the non-hierarchical approach attempts to `"fix the stove"`, while the hierarchical approach not only identifies the stove malfunction but actively seeks alternative solutions (`"Find other stove to heat apple juice"`), showing more sophisticated problem-solving strategies.

Expert Demonstration

w/o Hierarchy

Task Description: Your task is to find a(n) animal. First, focus on the thing. Then, move it to the red box in the kitchen.
Observation: This room is called the hallway. In it, you see: the agent, a substance called air, a picture. You also see: A door to the art studio (that is closed), A door to the bedroom (that is closed), A door to the greenhouse (that is closed), A door to the kitchen (that is closed), A door to the living room (that is closed), A door to the workshop (that is closed).
Action: open door to outside
Reward: 0.09
Observation: The door is now open.
Action: go to outside
Reward: 0.08
Observation: You move to the outside.
Action: look around
Reward: 0.0
Observation: This outside location is called the outside. Here you see: the agent, a substance called air, an axe, a crocodile egg, a fire pit (containing nothing), a fountain (containing a substance called water), the ground, a turtle egg, a baby wolf, a substance called wood. You also see: A door to the foundry (that is closed), A door to the greenhouse (that is closed), A door to the kitchen (that is open)
Action: focus on egg turtle
Reward: 0.5
Observation: You focus on the turtle egg.
Action: pick up egg turtle
Reward: 0.08
Observation: You move the turtle to the inventory.
Action: open door to kitchen
Reward: 0.0
Observation: The door is already open.
Action: go to kitchen
Reward: 0.0
Observation: You move to the kitchen.
Action: move egg turtle egg in inventory to red box
Reward: 0.17
Observation: You move the turtle to the red box.
done.

w/ Hierarchy

Task Description: Your task is to find a(n) animal. First, focus on the thing. Then, move it to the red box in the kitchen.
Observation: This room is called the hallway. In it, you see: the agent, a substance called air, a picture. You also see: A door to the art studio (that is closed), A door to the bedroom (that is closed), A door to the greenhouse (that is closed), A door to the kitchen (that is closed), A door to the living room (that is closed), A door to the workshop (that is closed).
Subtask: Find an animal and focus it
Action: open door to outside
Observation: The door is now open.
Action: go to outside
Observation: You move to the outside.
Action: look around
Observation: This outside location is called the outside. Here you see: the agent, a substance called air, an axe, a crocodile egg, a fire pit (containing nothing), a fountain (containing a substance called water), the ground, a turtle egg, a baby wolf, a substance called wood. You also see: A door to the foundry (that is closed), A door to the greenhouse (that is closed), A door to the kitchen (that is open)
Action: focus on egg turtle
High Reward: 0.75; **Low Reward:** 1.0
Observation: You focus on the turtle egg.
Subtask: Navigation to kitchen with the animal
Action: pick up egg turtle
Observation: You move the turtle to the inventory.
Action: open door to kitchen
Observation: The door is already open.
Action: go to kitchen
High Reward: 0.08; **Low Reward:** 1.0
Observation: You move to the kitchen.
Subtask: Move the animal to the red box in the kitchen
Action: move egg turtle egg in inventory to red box
High Reward: 0.17; **Low Reward:** 1.0
Observation: You move the turtle to the red box.
done.

Figure 6: Expert demonstration for w/o and w/ hierarchical structure in ScienceWorld.

Medium Trajectory Collected by In-distribution Sampling Strategy

w/o Hierarchy

Task Description: Your task is to boil apple juice. For compounds without a boiling point, combusting the substance is also acceptable. First, focus on the substance. Then, take actions that will cause it to change its state of matter.
Observation: This room is called the bathroom. In it, you see: the agent, substance called air, a bathtub, which is turned off. In the bathtub is: nothing. a glass cup (containing nothing), a picture, a sink, which is turned off. In the sink is: nothing. a toilet. In the toilet is: A drain, which is open, a substance called water. You also see: A door to the kitchen (that is closed).
Action: open door to kitchen
Reward: 0.0
Observation: .
Action: go to kitchen
Reward: 0.03
Observation: You move to the kitchen.
Action: look around
Reward: 0.0
Observation: This room is called the kitchen. In it, you see: a counter. On the counter is: a bowl (containing a red apple, a banana, an orange, a potato), a drawer. a cupboard. The cupboard door is closed. a freezer. The freezer door is closed. a fridge. The fridge door is closed. a glass jar (containing a substance called sodium chloride) a lighter a oven, which is turned off. The oven door is closed. a painting a sink, which is turned off. In the sink is: nothing. a substance called soap a stopwatch, which is deactivated. a stove, which is turned off. On the stove is: nothing. a table. On the table is: a glass cup (containing nothing). a thermometer, currently reading a temperature of 10 degrees celsius You also see: A door to the bathroom (that is open) A door to the hallway (that is closed) A door to the outside (that is closed).
Action: pick up thermometer
Reward: 0.0
Observation: You move the thermometer to the inventory.
Action: pick up metal pot
Reward: 0.0
Observation: You move the metal pot to the inventory.
Action: pick up apple juice
Reward: 0.0
Observation: No known action matches that input.
not done.

w/ Hierarchy

Task Description: Your task is to boil apple juice. For compounds without a boiling point, combusting the substance is also acceptable. First, focus on the substance. Then, take actions that will cause it to change its state of matter.
Observation: This room is called the bathroom. In it, you see: the agent, substance called air, a bathtub, which is turned off. In the bathtub is: nothing. a glass cup (containing nothing), a picture, a sink, which is turned off. In the sink is: nothing. a toilet. In the toilet is: A drain, which is open, a substance called water. You also see: A door to the kitchen (that is closed).
Subtask: Navigate to kitchen
Action: open door to kitchen
Observation: The door is now open.
Action: go to kitchen
High Reward: 0.03; **Low Reward:** 1.0
Observation: You move to the kitchen.
Subtask: Prepare tools for measuring temperature and boiling
Action: pick up thermometer
Observation: You move the thermometer to the inventory.
Action: pick up metal pot
High Reward: 0.0; **Low Reward:** 1.0
Observation: You move the metal pot to the inventory.
Subtask: Find apple juice and focus on it
Action: open cupboard
Observation: The cupboard is now open.
Action: focus on apple juice
High Reward: 0.0; **Low Reward:** 0.0
Observation: No known action matches that input.
Subtask: Find apple juice and focus on it
Action: open fridge
Observation: The fridge is now open.
Action: focus on apple juice
High Reward: 0.67; **Low Reward:** 1.0
Observation: You focus on the apple juice.
Subtask: Heat apple juice with stove
Action: move cup containing apple juice to stove
Observation: You move the glass cup to the stove.
Action: activate stove
High Reward: 0.02; **Low Reward:** 0.0
Observation: The stove is broken.
Subtask: Can't heat apple juice because stove is broken
Action: no action generated
High Reward: 0.0; **Low Reward:** 0.0
Observation: No known action matches that input.
not done.

Figure 7: Comparison of w/o and w/ hierarchical trajectories collected through in-distribution sampling in ScienceWorld.

Medium Trajectory Collected by Cross-task Generalization Sampling Strategy

w/o Hierarchy

Task Description: Your task is to boil apple juice. For compounds without a boiling point, combusting the substance is also acceptable. First, focus on the substance. Then, take actions that will cause it to change its state of matter.

Observation: This room is called the bathroom. In it, you see: the agent, substance called air, a bathtub, which is turned off. In the bathtub is: nothing. a glass cup (containing nothing), a picture, a sink, which is turned off. In the sink is: nothing. a toilet. In the toilet is: A drain, which is open, a substance called water. You also see: A door to the kitchen (that is closed).

Action: open door to kitchen

Reward: 0.0

Observation: .

Action: go to kitchen

Reward: 0.03

Observation: You move to the kitchen.

Action: pick up thermometer

Reward: 0.0

Observation: You move the thermometer to the inventory.

Action: pick up metal pot

Reward: 0.0

Observation: You move the metal pot to the inventory.

Action: open fridge

Reward: 0.0

Observation: The fridge is now open.

Action: focus on apple juice

Reward: 0.67

Observation: You focus on the apple juice.

Action: move cup containing apple juice to stove

Reward: 0.02

Observation: You move the glass cup to the stove.

Action: activate stove

Reward: 0.0

Observation: The stove is broken.

Action: fix the stove

Reward: 0.0

Observation: No known action matches that input

not done.

w/ Hierarchy

Task Description: Your task is to boil apple juice. For compounds without a boiling point, combusting the substance is also acceptable. First, focus on the substance. Then, take actions that will cause it to change its state of matter.

Observation: This room is called the bathroom. In it, you see: the agent, substance called air, a bathtub, which is turned off. In the bathtub is: nothing. a glass cup (containing nothing), a picture, a sink, which is turned off. In the sink is: nothing. a toilet. In the toilet is: A drain, which is open, a substance called water. You also see: A door to the kitchen (that is closed).

Subtask: Navigate to kitchen

Action: open door to kitchen

Observation: The door is now open.

Action: go to kitchen

High Reward: 0.03; **Low Reward:** 1.0

Observation: You move to the kitchen.

Subtask: Prepare tools for measuring temperature and boiling

Action: pick up thermometer

Observation: You move the thermometer to the inventory.

Action: pick up metal pot

High Reward: 0.0; **Low Reward:** 1.0

Observation: You move the metal pot to the inventory.

Subtask: Find apple juice and focus on it

Action: open cupboard

Observation: The cupboard is now open.

Action: focus on apple juice

High Reward: 0.0; **Low Reward:** 0.0

Observation: No known action matches that input.

Subtask: Find apple juice and focus on it

Action: open fridge

Observation: The fridge is now open.

Action: focus on apple juice

High Reward: 0.67; **Low Reward:** 1.0

Observation: You focus on the apple juice.

Subtask: Heat apple juice with stove

Action: move cup containing apple juice to stove

Observation: You move the glass cup to the stove.

Action: activate stove

High Reward: 0.0; **Low Reward:** 0.0

Observation: The stove is broken.

Subtask: Find other stove to heat apple juice

Action: focus on stove

High Reward: 0.0; **Low Reward:** 0.0

Observation: No known action matches that input.

not done.

Figure 8: Comparison of w/o and w/ hierarchical trajectories collected by cross-task generalization sampling in ScienceWorld.

Data Structure and Setups

The high-level and low-level training dataset structure follows a sequential format that captures the complete interaction trajectory:

Training Dataset Structure

High-Level Trajectory:
{high prompt, task description, obs 0, subtask 0, high reward 0 ... obs T-1, subtask T-1, high reward T-1, obs T}

Low-Level Trajectory:
{Low prompt, subtask 0, obs 0, action 0, low reward 1 ... obs c-1, action c-1, low reward c-1, obs c}

O2O task Setups

To evaluate the generalization capabilities of our method, we construct an O2O (Online-to-Offline) dataset covering three distinct domains: electrical, biology, and thermodynamics. Each domain contains one test task and multiple train tasks, as shown in Table 4. For the electrical domain, we have `"test-conductivity"` as the test task, along with three train tasks related to conductivity testing and power components. The biology domain features `"find-animal"` as the test task, accompanied by ten train tasks involving various biological concepts such as living/non-living identification, plant growth, and lifespan studies. In the thermodynamics domain, `"boil"` serves as the test task, supported by seven train tasks covering different aspects of state changes and chemical mixing processes. This setup provides a rigorous test of the agent's ability to transfer knowledge from trained tasks to novel but related domains in the ScienceWorld environment.

Table 4: Distribution of test and train tasks across electrical, biology, and thermodynamics domains

Domain	Task Name	Type
Electrical	test-conductivity	Test
	test-conductivity-of-unknown-substances	Train
	power-component	Train
	power-component-renewable-vs-nonrenewable-energy	Train
Biology	find-animal	Test
	find-living-thing	Train
	find-non-living-thing	Train
	find-plant	Train
	grow-fruit	Train
	grow-plant	Train
	identify-life-stages-1	Train
	identify-life-stages-2	Train
	lifespan-longest-lived	Train
	lifespan-longest-lived-then-shortest-lived	Train
	lifespan-shortest-lived	Train
Thermodynamics	boil	Test
	freeze	Train
	melt	Train
	change-the-state-of-matter-of	Train
	chemistry-mix	Train
	chemistry-mix-paint-secondary-color	Train
	chemistry-mix-paint-tertiary-color	Train
	use-thermometer	Train

Reward Design

- **ScienceWorld**: The agent receives a dense reward ranging from 0 to 1 at each step, reflecting the continuous progress in scientific experimentation tasks across 30 scenarios in 10 categories.
- **ALFWorld**: The agent receives a sparse binary reward (0 or 1), where 1 is given only upon successful completion of household navigation and manipulation tasks, and 0 otherwise.

- **High-level Reward**: The high-level policy accumulates environmental rewards upon the completion of subtasks by the low-level policy, reflecting the agent's progress in achieving the overall objective.
- **Low-level Reward**: The low-level policy receives binary rewards from the high-level policy, indicating whether a subtask is successfully completed or not.

Appendix C. Training and Evaluation setups

Models

We build our method on three open-source language models: 1) `Mistral-7B` (Jiang et al., 2023), the Mistral-7B-Instruct-v0.2 version. 2) `Gemma-7B` (Team et al., 2024), the Gemma-1.1-7B-it version. 3) `Llama-3-8B` (Meta, 2024), the Meta-Llama-3-8B-Instruct version. We employ LoRA for parameter-efficient fine-tuning of all language models.

Baselines

We compare GLIDER against various strong baselines: 1) `ReAct` (Yao et al., 2023b), a pioneering approach that incorporates CoT prompting in decision-making tasks through a structured Thought-Action-Observation loop. 2) `Reflexion` (Shinn et al., 2023), an advanced prompt-based framework that enhances agent decision-making through self-reflective verbal feedback. 3) `SwiftSage` (Lin et al., 2023), a dual-process theory-based cognitive framework that integrates the strengths of behavior cloning and prompting for complex interactive reasoning and action-planning tasks. 4) `NAT` (Wang et al., 2024), a fine-tuning approach that enables LLMs to learn from failure trajectories through quality control and fine-tuning strategies. 5) `ETO` (Song et al., 2024), an iterative optimization framework between exploring the environment to collect contrastive trajectory pairs and fine-tuning the LLM policy using DPO (Rafailov et al., 2023).

Hyperparameter

we employ LoRA for parameter-efficient fine-tuning for all language models. During the SFT phase, we train for 5 epochs with a batch size of 32, using the AdamW optimizer with a learning rate of $1e-4$. The detailed hyperparameters are shown in the following table 5. Policy are evaluated on both seen and unseen tasks across two benchmarks. For the ORL phase, we train for 4 epochs with different learning rates for the actor ($1e-5$) and critic ($1e-4$) networks. The target critic network is updated using soft updates with $\tau = 0.2$, and we set the advantage weighted factor λ to 0.99. The training data consists of expert and medium data in a 1:2 ratio.

Table 5: Hyperparameters used for GLIDER.

Hyperparameter	Value	Hyperparameter	Value
batch size	64	temperature	0.7
batch size per device	2	advantage weighted factor λ	0.99
gradient accumulation steps	8	soft update τ	0.2
actor learning rate	1×10^{-5}	discount factor γ	0.99
critic learning rate	1×10^{-4}	sft epochs	5
sft learning rate	1×10^{-4}	orl epochs	4
lora r	16	data mixture radio	1:2
lora alpha	32	warmup ratio	0.03
lora dropout	0.05	max new tokens	32

Appendix D. Prompts and Case Study

We illustrate the prompts used in our paper, where High-Level and Low-Level prompts establish hierarchical control between different agents, while Check Subtask Complete Prompt enables the high-level agent to evaluate subtask completion by the low-level agent.

Prompts

High-Level Prompt:
You are a high-level planner. Based on the state (task description, group action and current observation), please generate a clear and simple subtask.

Low-Level Prompt:
You are a low-level action executor. Based on the current subtask and observation, please generate a executable action and determine if the subtask is completed (true/false).

Check Subtask Complete Prompt:
Determine if the low-level actions successfully completed the given subtask by high-level:
Subtask: [subtask]
Initial observation: [initial obs]
Actions: [action sequence]
Final observation: [final obs]
Output only a single digit:
True if the actions successfully completed the subtask
False if the actions failed to complete the subtask
Give the ”True” or ”False”:

To demonstrate the advantages of hierarchical decomposition, we present a case study show in Figure. 9 comparing two different tasks: freezing apple juice and boiling water. The highlighted portions reveal two types of structural similarities: identical subtasks (highlighted in yellow) that can be directly reused, and analogous subtasks (highlighted in red) that share similar underlying patterns despite different objectives. For example, both tasks contain the identical subtask `"Prepare temperature and metal pot"`, while `"Monitor apple juice until frozen"` and `"Monitor water until boiling"` represent analogous patterns of state monitoring and waiting. This hierarchical approach not only enables direct subtask reuse, but also allows the planner to recognize and adapt similar strategic patterns across different tasks, thereby reducing the planning complexity and improving efficiency.

Case Study

freeze apple juice

Task Description: Your task is to melt ice. First, focus on the substance. Then, take actions that will cause it to change its state of matter.
Observation: This room is called the greenhouse. In it, you see: the agent, a substance called air, a bee hive. The bee hive door is closed, a jug (containing nothing), a shovel, sink, which is turned off.
Subtask: Navigate to kitchen
Action: open door to hallway
Observation: The door is now open.
Action: go to hallway
Observation: You move to the hallway.
Action: open door to kitchen
Observation: You move to the kitchen.
Subtask: Prepare temperature and metal pot
Action: pick up thermometer
Observation: You move thermometer to inventory.
Action: open cupboard
Observation: The cupboard is now open.
Action: pick up metal pot
Observation: You move metal pot to inventory.
Subtask: Find apple juice then focus it
Action: open fridge
Observation: The fridge is now open.
Action: focus on apple juice
Observation: You focus on the apple juice.
Subtask: Move apple juice to metal pot then freeze it
Action: pour cup containing apple juice into metal pot
Observation: You pour the apple juice into metal pot.
Action: pick up metal pot
Observation: You move the metal pot to the inventory.
Action: open freezer
Observation: The freezer is now open.
Action: move metal pot to freezer
Observation: You move the metal pot to the freezer.
Subtask: Monitor apple juice until frozen
Action: examine apple juice
Observation: orange juice
Action: use thermometer in inventory on apple juice
Observation: The temperature is 2 degrees celsius
Action: wait
Observation: You decide to wait for 10 iterations.
Action: examine orange juice
Observation: solid orange juice
done.

boil water

Task Description: Your task is to boil water. For compounds without boiling point, combusting the substance is acceptable. First, focus on the substance. Then, take actions that will cause it to change its state of matter.
Observation: This room is called the kitchen. In it, you see: a counter. On the counter is: a bowl (containing a red apple, a banana, an orange, a potato), a drawer. a cupboard. The cupboard door is closed. a freezer. The freezer door is closed. a fridge. The fridge door is closed.
Subtask: Prepare temperature and metal pot
Action: pick up thermometer
Observation: You move thermometer to inventory.
Action: open cupboard
Observation: The cupboard is now open.
Action: pick up metal pot
Observation: You move metal pot to inventory.
Subtask: Fill metal pot with water and focus on it
Action: move metal pot to sink
Observation: You move the metal pot to the sink.
Action: activate sink
Observation: The sink is now activated.
Action: deactivate sink
Observation: The sink is now deactivated.
Action: focus on substance in metal pot
Observation: You focus on the water.
Subtask: Heat water with stove
Action: move metal pot to stove
Observation: You move the metal pot to the stove.
Action: activate stove
Observation: The stove is now activated.
Subtask: Monitor water until boiling
Action: examine substance in metal pot
Observation: a substance called water
Action: use thermometer in inventory on substance in metal pot
Observation: The temperature is 68 degrees celsius
Action: wait2
Observation: You decide to wait for 20 iterations.
Action: examine substance in metal pot
Observation: a substance called steam.
done.

Figure 9: Hierarchical decomposition reveals shared subtask patterns across two example tasks in ScienceWorld.