Runaway is Ashamed, But Helpful: On the Early-Exit Behavior of Large Language Model-based Agents in Embodied Environments

Anonymous ACL submission

Abstract

Agents powered by large language models (LLMs) have demonstrated strong planning and decision-making capabilities in complex embodied environments. However, such agents often suffer from inefficiencies in multi-turn interactions, frequently trapped in repetitive loops or issuing ineffective commands, leading to redundant computational overhead. Instead of relying solely on learning from trajectories, we take a first step toward exploring the early-exit behavior for LLM-based agents. We propose two complementary approaches, ❶ an **intrinsic** method that injects exit instructions during generation, and ❷ an **extrinsic** method that verifies task completion to determine when to halt an agent's trial. To evaluate early-exit mechanisms, we introduce two metrics: one measures the reduction of **redundant steps** as a positive effect, and the other evaluates **progress degradation** as a negative effect. Experiments with 4 different LLMs across 5 embodied environments show significant efficiency improvements, with only minor drops in agent performance. We also validate a practical strategy where a stronger agent assists after an early-exit agent, achieving better performance with the same total steps. We will release our code to support further research.

1 Introduction

Large Language Models (LLMs, Achiam et al., 2023) have shifted the paradigm from merely responding to user inputs to tackling more complex tasks within interactive environments such as household settings (Shridhar et al., 2021), virtual worlds (Park et al., 2023), and games (Hu et al., 2024). LLM-based agents serve as intelligent controllers, capable of perceiving environments, executing actions, and adapting through feedback (Wang et al., 2024; Luo et al., 2025). Previous studies show that structured workflows—such as reasoning before acting (Yao et al., 2023), predicting future states (Fu et al., 2025b), and learning from high-quality trajectories (Chen et al., 2024b; Song et al., 2024)—can improve performance within a single trial. When agents do fail, post-hoc approaches such as Reflexion (Shinn et al., 2023), AutoPlan (Ouyang and Li, 2023), and ExpeL (Zhao et al., 2024) enable them to learn from failures and replan more effective solutions in subsequent trials.

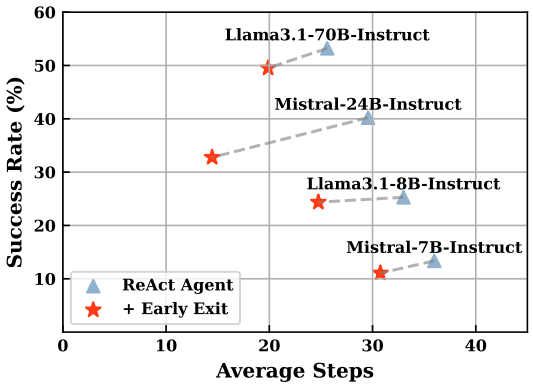


Figure 1: **Early-exit behavior of different LLM-based agents** in embodied environments. While early termination slightly reduces the success rate, it significantly decreases the average number of interaction steps, indicating improved efficiency.

However, a key limitation of LLM-based agents remains underexplored: *they often fail to recognize when a goal is too difficult or when they are stuck*. Prior work shows that agents may repeat the same errors in unproductive loops without meaningful actions or self-correction (Fu et al., 2025a), leading to unnecessary computational overhead. This issue becomes even more critical in real-world settings, where repeated mistakes by embodied agents can waste energy, cause wear-and-tear, or even damage physical objects in the environment. Therefore, incorporating built-in self-awareness mechanisms can help agents detect when progress has stalled, enabling early self-reflection and adjustment.

To this end, we take the first step by investigating

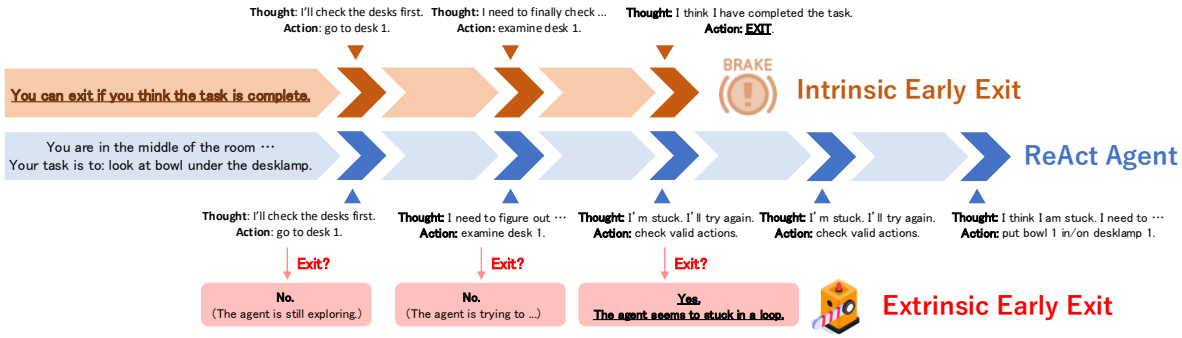


Figure 2: **A comparative overview of our proposed Intrinsic and Extrinsic Early Exit** with ReAct Agent. The intrinsic approach injects an exit instruction to guide the agent to self-terminate, while the extrinsic approach uses a verification module to determine whether to exit based on the current state.

the *early-exit* behavior of LLM-based agents. As shown in Figure 2, we propose two complementary strategies: ❶ **Intrinsic Early Exit**, which injects exit instructions directly into the agent's prompts to encourage self-recognition of when to halt; and ❷ **Extrinsic Early Exit**, which introduces an external verification module that monitors the interaction status and outputs a binary (YES/NO) decision to control whether the agent should continue.

In addition to using success rate and progress rate (Chang et al., 2024) to evaluate agent performance, we propose two new metrics to assess the impact of the early-exit mechanism. *Redundancy Steps* quantifies the positive effect by measuring reductions in unnecessary interactions, while *Progress Degradation* captures the potential negative impact, indicating cases where exit early may interrupt or reverse meaningful progress.

We conduct experiments on 5 datasets spanning over 400 environments and find that the early-exit mechanism significantly improves efficiency, with only a minor drop in task success and progress rates, as shown in Figure 1. We also propose a practical use of early-exit behavior: Once the agent exits early, a stronger agent reflects on the state and continues exploration, achieving improved performance within the same total steps.

Our contributions are three-fold:

- We present the first investigation into early-exit behavior in LLM-based agents, proposing two strategies that enable agents to develop self-awareness and terminate execution without external intervention.
- We introduce two complementary metrics to evaluate the effectiveness of early exit. These metrics can serve as standardized tools for assessing agent behavior and guiding the selection of optimal exiting strategies.
- Our proposed methods generalize across various LLM-based agents and task settings. We further demonstrate the practical value of our approach by introducing post-trial strategies that leverage stronger agents to enhance overall performance.

This study is an initial step toward exploring early-exit behavior in LLM-based agents. Our approach encourages agents to make efficient decisions, avoid unnecessary interactions, and achieve a trade-off between efficiency and task performance.

2 Approach

2.1 Task Formulation

Embodied Environments In embodied environments, an agent interacts with the world through actions and receives feedback from the environment. This interaction can be modeled as a special case of a Partially Observable Markov Decision Process (POMDP), defined by an instruction space U, state space S, action space A, observation space O, and a transition function $T : S \times A \to S$.

LLM-based Agents In this work, we focus on text-based environments, where the instruction, action, and observation spaces are all expressed in natural language. The agent is provided with an instruction u, which includes a description of the task and environment, as well as the goal to be achieved. At each time step t, the agent, guided by a policy π_θ (typically an LLM with parameters θ), must decide on the next action a_t based on the trajectory history e_t. This decision-making process

is formalized as:

$$a_t \sim \pi_\theta(\cdot \mid e_t, u), \tag{1}$$

where $e_t = (a_1, o_1, \ldots, a_{t-1}, o_{t-1})$ denotes the full trajectory up to time t, including previous actions and observations. In this way, the agent continually explores the environment, using feedback from observations to inform its next actions, until the task is completed or a predefined maximum number of steps is reached.

2.2 Dynamic Early Exit

We propose two simple but effective early-exit strategies, *Intrinsic Early Exit* and *Extrinsic Early Exit*, that enable the agent to terminate its interaction when appropriate.

Intrinsic Early Exit This strategy modifies the behavior of LLM agent by appending a natural language prompt that allows it to terminate the interaction with the environment when deemed necessary. The exit instruction can be formulated as:

$$u_{\text{intrinsic}} = \text{concat}(u, u_{\text{exit}}). \tag{2}$$

In this way, the LLM may develop an intention to termiate based on the additional instruction u_{exit}, leading to different actions and trajectories. As shown in Figure 2, the agent is prompted with an instruction to exit once the task is complete. After examining the relevant objects, the agent generates an "EXIT" action to terminate the interaction.

Extrinsic Early Exit This strategy introduces a verification module v_θ, which shares the same LLM backbone. The verification module operates after each action and observation, evaluating whether the agent should continue the task. It outputs a binary decision: "YES" to exit or "NO" to continue execution. Specifically, it functions as follows:

$$u_{\text{extrinsic}} = \text{concat}(u, u_{\text{exit}}), \tag{3}$$

$$v_\theta(\cdot \mid e_t, u_{\text{extrinsic}}) \in \{0, 1\}. \tag{4}$$

The agent is verified periodically every k steps. In our experiments, we set $k = 1$[1]. As shown in Figure 2, the verification module detects that the agent is stuck and triggers an early exit, effectively avoiding further repetitive steps.

2.3 Evaluation

Typically, the performance of agents in embodied environments is evaluated using Success Rate and Progress Rate. To intuitively demonstrate the behavior of the early-exit mechanism on LLM-based agents, we propose two complementary metrics that capture both its positive and negative effects. These metrics are defined as follows:

Success Rate (SR) The environment is marked as successful if the agent completes the given task, typically when it reaches a predefined latent state that signifies task completion. A higher success rate indicates that the agent is more effective at solving environments under the same task.

Progress Rate (PR) Progress Rate, proposed by Chang et al. (2024), quantifies the extent to which an agent advances toward the task goal, making it particularly valuable for evaluating incremental improvements. In embodied environments, the task goal is decomposed into a sequence of subgoals $G = [g_1, \cdots, g_K]$, where each subgoal contributes progressively to task completion. At each time step t, the progress is defined as:

$$r_t = \max_{i, 0 \le i \le t} \left(\frac{1}{K} \sum_{k=1}^{K} f(s_i, g_k) \right), \tag{5}$$

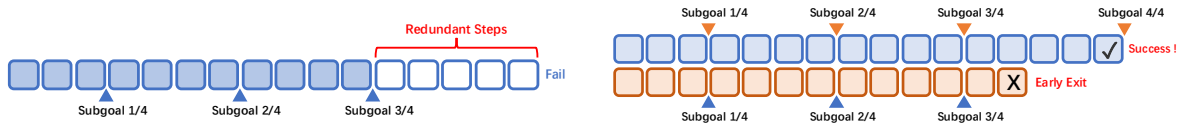
where $f(s_i, g_k) \in \{0, 1\}$ is a binary indicator function that evaluates whether the agent state s_i satisfies subgoal g_k, typically determined via regular-expression-based matching. PR offers a more fine-grained and informative evaluation of agent behavior than binary success metrics alone.

New Metric 1: Redundancy Steps (RS) The primary purpose of introducing the early-exit mechanism is to reduce redundant steps in the agent's interaction with the environment. As illustrated in Figure 3(a), after completing subgoal 3 out of 4, the agent continues exploring unnecessarily for 5 additional steps before ultimately failing. Early-exit can mitigate this issue while maintaining the same level of progress. Let n_{total} denote the total number of steps in the trajectory, and n_{subgoal} be the index of the last step that achieves a new subgoal. The *Redundancy Steps* is defined as:

$$\text{RS} = n_{\text{total}} - n_{\text{subgoal}}. \tag{6}$$

For trivial cases, $\text{RS} = n_{\text{total}}$ if the agent fails to complete any subgoal (i.e., $\text{PR} = 0$). if the agent successfully completes the entire task, $\text{RS} = 0$, meaning that all steps are considered useful.

[1]We set $k = 1$ to enable timely detection in our experiments. In practice, larger values (e.g., $k = 2$–5) can be used to reduce computational overhead.



(a) Redundancy Steps (b) Progress Degradation

Figure 3: **An overview of the proposed metrics**. *Redundancy Steps* measures the number of redundant steps. *Progress Degradation* measures task progress loss via reduced subgoal completion.

New Metric 2: Progress Degradation (PD) The agent may also negatively impact agent performance by prematurely terminating trajectories that might have led to further progress. This can suppress the agent's potential, causing missed subgoals or converting potentially successful trials into failures. To quantify this loss, we define *Progress Degradation* as:

$$\text{PD} = \max(\text{PR}_{\text{ref}} - \text{PR}_{\text{exit}}, 0), \quad (7)$$

where PR_{ref} denotes the progress rate without exit, while PR_{exit} is the progress rate when early-exit is applied[2]. As shown in Figure 3(b), the agent exits 3 steps early, leaving an otherwise successful environment unfinished with only 75% progress, resulting in a 25% loss in progress. *Progress Degradation* ranges from 0 (no degradation) to PR_{ref} (complete loss of progress). A higher PD indicates greater performance loss. In the trivial case, $\text{PD} = 0$ implies no degradation, while $\text{PD} = \text{PR}_{\text{ref}}$ indicates complete progress failure (e.g., all environments terminate at the first step).

3 Experimental Setup

Datasets We evaluate our methods across 3 embodied environments and 2 gaming environments. For embodied environments, **AlfWorld** (Shridhar et al., 2021) includes 134 household tasks that require agents to explore their surroundings and complete instructions such as "Look at bowl under the desklamp." **ScienceWorld** (Wang et al., 2022) simulates a total of 90 scientific experiments in an interactive setting, such as "measure the melting point." **BabyAI** (Chevalier-Boisvert et al., 2019) is a 20x20 grid-based environment where agents must navigate and interact with objects to accomplish 112 defined goals. We also consider two gaming environments. **Jericho** (Hausknecht et al., 2020) comprises 20 text-based fictional worlds, which we adapt using the setup from Chang et al. (2024) to be completed within 15 subgoals. **PDDL** represents a suite of strategic planning tasks defined in the Planning Domain Definition Language (Vallati et al., 2015). Following Chang et al. (2024), we include four distinct games, namely, 60 unique environments for evaluation.

LLMs To ensure reproducibility, we evaluate four open-source large language models with varying parameter sizes. From the LLaMA 3.1 series[3] (Grattafiori et al., 2024), developed by Meta, we use two instruction-tuned models: the 8B version (*Llama3.1-8B-Instruct*) and the 70B version (*Llama3.1-70B-Instruct*), with the latter quantized using 4-bit AWQ (Lin et al., 2024) for efficient inference. In addition, we test two models from the Mistral family[4] (Jiang et al., 2024): *Mistral-7B-Instruct* (v0.3) and *Mistral-24B-Instruct* (Mistral-Small-Instruct-2409).

Prompts We adopt ReAct-style (Yao et al., 2023) prompting to enable LLM-based agents to interact effectively with the environment. Following Song et al. (2024), we format the interaction prompt as a multi-turn conversation, including an in-context example for each task. For early-exit instructions, we explore prompt variants with varying strictness levels (see in Appendix B), aligning the strategy with specific LLMs.

Hyperparameters For all experiments, we set the temperature to 0.1 and limit each turn's response to a maximum of 256 tokens.

Device and Platform All experiments are conducted on two NVIDIA A100 GPUs with 80GB of memory each. We deploy the models using VLLM (Kwon et al., 2023) for distributed inference and access them through OpenAI-compatible chat completion APIs (Achiam et al., 2023). Evaluation is performed using AgentBoard (Chang et al., 2024), measuring both success rate and progress rate.

[2]Progress degradation is only meaningful when compared against a reference baseline.

[3]https://huggingface.co/meta-llama

[4]https://huggingface.co/mistralai

Setting		ALFWorld					BabyAI					ScienceWorld				
Int.	**Ext.**	SR↑	PR↑	RS↓	PD↓	Steps↓	SR↑	PR↑	RS↓	PD↓	Steps↓	SR↑	PR↑	RS↓	PD↓	Steps↓
Llama3.1-8B-Instruct																
-	-	23.1	45.2	13.8	-	33.4	41.1	54.6	8.2	-	27.1	8.9	37.3	16.5	-	38.5
✓	✗	14.2	38.3	4.1	14.1	15.9	41.1	54.3	5.3	10.5	25.6	7.8	32.6	13.5	11.1	32.3
✗	✓	20.9	38.3	2.5	16.3	9.6	16.1	25.4	1.6	30.5	6.6	7.8	29.5	4.6	13.9	10.5
✓	✓	21.6	44.4	4.5	9.1	16.8	46.4	57.3	6.5	6.6	25.1	7.8	34.5	13.0	9.4	32.3
Llama3.1-70B-Instruct																
-	-	76.1	81.1	2.3	-	19.0	49.1	62.8	8.8	-	26.4	34.4	67.5	15.7	-	31.4
✓	✗	61.2	67.4	1.4	17.4	13.8	36.6	53.1	7.4	18.0	18.2	18.9	59.8	8.3	12.3	21.2
✗	✓	70.2	79.3	1.4	8.7	13.4	42.0	59.3	4.5	12.9	13.3	27.8	63.6	6.0	9.0	17.4
✓	✓	80.6	84.0	1.5	5.8	17.0	40.2	57.8	6.8	13.1	19.9	27.8	64.6	8.7	10.6	22.8
Mistral-7B-Instruct																
-	-	20.9	40.4	13.9	-	34.8	17.0	21.7	5.6	-	34.0	2.2	16.6	15.3	-	39.2
✓	✗	14.9	36.3	5.9	15.9	23.9	16.1	25.9	7.7	5.6	32.0	2.2	18.4	15.2	3.4	36.3
✗	✓	11.2	32.5	8.9	16.7	24.9	10.7	18.2	5.0	12.0	27.9	1.1	15.4	7.8	3.4	17.2
✓	✓	17.2	36.1	12.1	11.4	32.7	16.1	22.4	5.7	4.9	33.5	2.2	18.2	15.4	1.9	38.0
Mistral-24B-Instruct																
-	-	58.2	71.6	5.0	-	25.9	49.1	60.9	7.0	-	25.5	15.6	42.5	16.0	-	36.9
✓	✗	31.3	51.7	4.0	26.0	17.4	40.2	51.5	8.2	19.4	27.0	11.1	40.7	11.3	18.9	31.7
✗	✓	57.5	70.7	4.5	10.8	20.5	37.5	50.1	3.2	16.3	13.3	3.3	23.3	4.3	20.7	9.5
✓	✓	57.5	74.3	5.7	10.5	25.7	35.7	53.9	10.5	19.6	28.4	12.2	39.5	12.0	18.0	35.2

Table 1: **Performance comparison of two early-exit approaches**—Extrinsic (Ext.) and Intrinsic (Int.)—vs. the ReAct baseline across four LLMs in three embodied environments. Red indicates **negative** impact (e.g., performance drop or progress degradation), while Green shows **positive** effects (e.g., reduced redundancy). Metrics: SR (Success Rate), PR (Progress Rate), RS (Redundant Steps), PD (Progress Degradation), and Steps (Average Steps).

4 Main Results

We experiment on 3 embodied environments and 2 gaming environments, and report results in Table 1 and Table 2, respectively. We can see that:

(i) Early-exit mechanisms significantly reduce redundant steps. Across all three embodied environments, baseline methods exhibit substantial redundancy ("RS") in their thought-action sequences. For example, *LLama3.1-8B-Instruct* averages 13.8 unnecessary steps out of 40 in Alfworld. Almost all early exit mechanisms are able to reduce the redundancy, by approximately 50% to 70%, leading to a notable increase in overall efficiency. A similar trend is observed in the average steps ("Steps"), decreasing alongside the reduction in redundant steps, further highlighting the effectiveness of the early-exit mechanism in improving task efficiency.

(ii) Minor performance drop in success and progress rates. While early exit improves efficiency, it inevitably causes slight reductions in both success and progress rates. The observed progress degradation ("PD") further confirms this trade-off. However, for all four tested LLMs, certain early exit strategies yield minimal performance loss. For example, using the extrinsic ("Ext.") method on *Llama3.1-70B-Instruct*, the progress rate drops by under 2%, 3%, and 4% in ALFWorld, BabyAI, and ScienceWorld, respectively. This shows that appropriate early exits can greatly improve efficiency with negligible performance impact.

(iii) LLMs show varying preferences for early exit strategies. LLMs respond differently to the same early exit approach. For example, the intrinsic ("Int.") early exit performs better for *Mistral-7B-Instruct*, whereas it significantly degrades the performance of *Mistral-24B-Instruct*. Conversely, *Mistral-24B-Instruct* benefits more from the extrinsic method ("Ext."). This is possibly because the larger Mistral LLMs is more sensitive to intrinsic cues, resulting in premature termination, whereas extrinsic method provide more stable exit signals.

(iv) Combining intrinsic and extrinsic early exit maximizes performance retention. We explore a hybrid strategy that first applies extrinsic verification to detect potential exit, then applies the intrinsic method to confirm termination. While this increases the number of steps and reduces efficiency, it achieves the best performance preser-

Setting		PDDL					Jericho				
Int.	**Ext.**	SR↑	PR↑	RS↓	PD↓	Steps↓	SR↑	PR↑	RS↓	PD↓	Steps↓
Llama3.1-8B-Instruct											
-	-	11.7	29.9	11.8	-	38.3	5	27.3	13.8	-	36.5
✓	✗	6.7	30.5	5.5	6.1	31.4	5	26.8	10.4	9	37.7
✗	✓	1.7	4.4	1.1	25.9	4	0	7.5	1.9	19.8	6.8
✓	✓	8.3	31.4	6.7	6.1	32.3	10	31.8	13.1	10.5	33.3
Llama3.1-70B-Instruct											
-	-	45	62.2	6.5	-	31.1	35	55.9	11.9	-	32.3
✓	✗	41.7	64.8	4.4	5.8	28.2	25	41.5	7.8	23.1	29.8
✗	✓	43.3	63.5	2.5	4.9	23	20	38.5	7.5	19.8	21.7
✓	✓	38.3	61.9	6.8	8.1	29.7	20	41.5	10.1	21.1	29.4
Mistral-7B-Instruct											
-	-	0	9.7	12.2	-	40	0	11.7	16.1	-	38.4
✓	✗	1.7	12.1	8.9	5	30.3	0	6.9	9.9	4.8	30.2
✗	✓	3.3	13.8	6.6	4.6	20.9	0	9	7.5	6	26.1
✓	✓	3.3	12.9	9.6	4.2	35.6	0	12	12.2	3.5	36.6
Mistral-24B-Instruct											
-	-	13.3	27.4	7.5	-	37	15	43.8	19.2	-	37.3
✓	✗	13.3	33.3	8.6	8.5	34.7	10	33.8	14.4	12.7	32.7
✗	✓	10.0	24.3	3.8	7.9	16.2	5	29.5	9.3	18.2	20.9
✓	✓	11.7	32.0	9.9	7.8	36.1	10	27.4	12.9	20.5	37.0

Table 2: **Performance comparison of two early-exit settings** across four LLMs in game environments. Red indicates negative impact, while Green shows positive effects (e.g., reduced redundancy). Metrics: SR (Success Rate), PR (Progress Rate), RS (Redundant Steps), PD (Progress Degradation), and Steps (Average Steps).

vation ("Int.✓+ Ext.✓"). Notably, it even slightly improves performance on *Llama3.1-70B-Instruct* and *Mistral-24B-Instruct*, possibly due to diversity behavior introduced by prompt modification.

(v) Early-exit strategy generalizes to gaming environments. As shown in Table 2, applying early-exit in gaming environments yields similar trends, but smaller gains in efficiency and minor performance changes compared to embodied tasks. Redundancy reduction is less pronounced (generally below 50%), and the drops in performance are marginal, except for *Mistral-7B-Instruct*, occasionally showing improvement. This may be due to: 1) the longer trajectories in gaming environments, which lead to lower baseline success rates (e.g., below 20% for most LLMs except *Llama3.1-70B-Instruct*) and greater sensitivity to prompt variations; and 2) ambiguous subgoal definitions, allowing multiple valid strategies and reducing consistency in progress measurement.

5 Analysis

5.1 Interpretation of Efficiency Metrics

We illustrate how Redundancy Steps (RS) and Progress Rate (PR) complement each other in measuring the early-exit behavior in Figure 4.

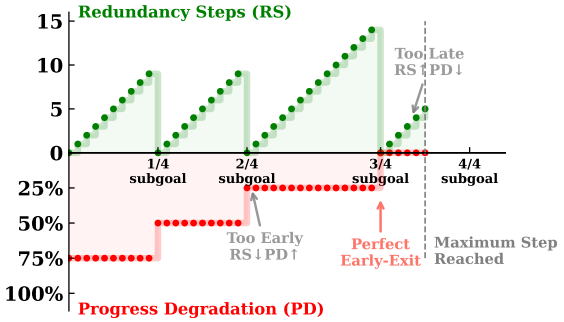


Figure 4: **Redundant Steps and Progress Degradation** measured in a failure case with 3 out of 4 subgoals completed. The metrics vary as the early-exit mechanism is triggered at different steps.

Perfect Early-Exit Scenario The ideal early exit scenario ("Perfect Early Exit") occurs when both RS and PR are zero, meaning no redundant steps and no progress loss. However, this ideal is rarely achievable across all environments in practice.

Too-Early Scenarios If the early exit mechanism triggers too early ("Too Early"), it may reduce redundant steps but significantly impair progress. This is evident in the result of the external early exit of *Llama3.1-8B-Instruct* on BabyAI, where early termination yields a low RS but a high PD of 30.5.

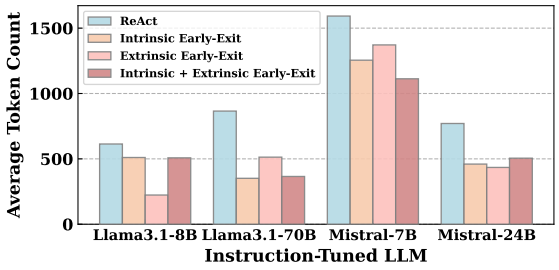


Figure 5: **Comparison of the average token cost** for one environment using different early-exit mechanisms.

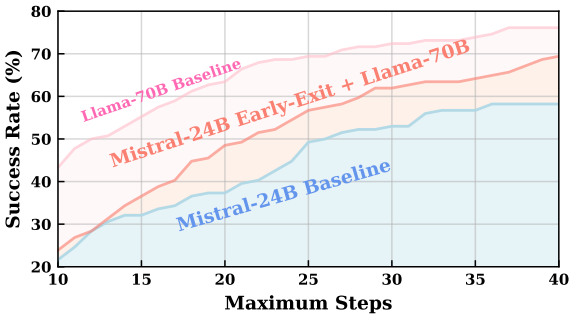


Figure 6: **Performance comparison under different max step limits** using strong-agent assistance with an early-exit weak agent, compared to baseline agents. *Mistral-24B-Instruct* ("Mistral-24B") is used as the weak agent, and *Llama-3.1-70B-Instruct* ("Llama-70B") as the strong agent.

Too-Late Scenarios Conversely, if the early exit mechanism triggers too late ("Too Late"), PD remains low but RS stays high. This is seen in *Mistral-24B-Instruct*, when using both intrinsic and extrinsic early exit methods fail to reduce RS.

Takeaways Neither too-early nor too-late exits are optimal in practice. Our results highlight the importance of selecting appropriate early exit settings for each LLM to balance RS and PR effectively.

5.2 Inference Cost

To further validate the efficiency improvements achieved by the early-exit mechanism, in addition to reporting the average number of execution steps in the main results, we also examine the average token cost for each environment, which directly reflects the computational resource usage. As shown in Figure 5, the early-exit approach consistently reduces the number of tokens compared to ReAct across all four tested LLMs. It is worth noting that, in our extrinsic early-exit approach, the verification module generates only a simple "YES" or "NO" response. As a result, it has a negligible impact on the overall token cost.

6 Practical Implications

6.1 Motivation

A key advantage of our proposed early-exit mechanism is that agents capable of recognizing failure can proactively terminate and seek assistance, leading to more efficient problem-solving. This aligns with realistic application scenarios, where humans may intervene directly or request help via a central server. In contrast, agents without early-exit continue until the step limit, often wasting valuable interactions and failing to complete the task.

To illustrate this, we simulate a practical scenario in embodied environments, where **a weaker agent exits early from challenging environments and requests assistance from a stronger agent**.

6.2 Setting

Dataset We use ALFWorld (Shridhar et al., 2021) as our test set, which is a typical embodied environment with 134 different tasks.

Models We use *Mistral-24B-Instruct* as the weak agent which achieves a 58.2% success rate under a ReAct-style format and 57.5% when paired with an external early-exit mechanism (seen in Table 1), and *Llama3.1-70B-Instruct* as the strong agent.

Setup In baseline, the weaker agent executes up to 40 steps regardless of progress. With the extrinsic early-exit mechanism, it can terminate early and hand over control to a stronger agent, which replans and continues within the remaining steps.

6.3 Experiment

Result As shown in Figure 6, early exit followed by strong agent assistance yields over a 10% improvement in success rate within the same 40-step budget, demonstrating the effectiveness of reallocating interaction steps to a more capable agent.

Case Study Figure 7 visualizes environments impacted by early-exit, where successful environments by both early-exit and baseline are ignored. Around 15 environments (e.g., #3, #12) were completed with strong-agent help. Of these, 7 environments (e.g., #12, #19) were not completed by the baseline within 40 steps but were solved with early exit and assistance. Only 2 cases (#29, #61) were prematurely exited but solved by baseline.

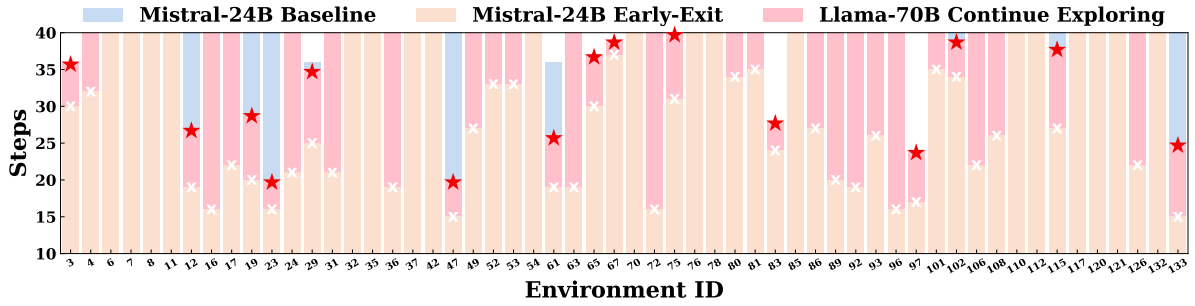


Figure 7: **Case study of failure environments under the early-exit approach** in ALFWorld. Different colors indicate the contributions of various strategies. "x" marks the exit step for each environment, and ★ indicates completion by the stronger agent (*Llama3.1-70B-Instruct*).

Some tasks (e.g., #4, #36) remained unsolved by both agents but benefited from reduced wasted computation. These results clearly highlight the efficiency improvements brought by early-exit, especially when supported by stronger agents.

7 Related Work

LLM-based Agents LLM-based agents are central to many tasks and show strong practical potential. Some approaches, like ETO (Song et al., 2024) and AgentFLAN (Chen et al., 2024b), improve performance through expert trajectory training, achieving better generalization. Others, such as ReAct (Yao et al., 2023), PreAct (Fu et al., 2025b), and StateFlow (Wu et al., 2024), focus on prompt design to enhance chain-of-thought (CoT, Wei et al., 2022) reasoning. While effective, these methods often neglect efficiency, especially in failure cases. Complementary post-hoc strategies—like self-reflection (Shinn et al., 2023), trajectory revision (Ouyang and Li, 2023), and experience extraction (Zhao et al., 2024)—help refine future behavior but only after trials conclude. We propose early-exit approaches that improve efficiency and demonstrate the practical benefits of leveraging stronger agents and post-hoc strategies.

Dynamic Early Exit Dynamic early exit is an adaptive inference strategy originally introduced in pre-trained language models to reduce computational cost and latency by skipping certain layers during inference (Zhou et al., 2020; Sun et al., 2022). Recent work extends this concept to LLMs to address the issue of excessively long and unpredictable generations. Yang et al. (2025) applies early exit mechanism to truncate outputs at appropriate reasoning steps, thereby mitigating the "overthinking" problem in LLMs (Chen et al., 2024a). In this work, we apply early exit to LLM-based agents in embodied environments, proposing an efficient and robust method adaptable to various agents, along with metrics to assess performance.

Agent Verification and Evaluation Traditional benchmarks like AgentBench (Liu et al., 2024) assess overall agent performance using metrics such as reward or success rate. AgentBoard (Chang et al., 2024) improves transparency with human-annotated subgoals for process-level evaluation. A growing line of work explores using agents themselves as evaluators, extending ideas from text generation (Zheng et al., 2023; Lu et al., 2024), code evaluation (Chen et al., 2024b). For example, Pan et al. (2024) explore using agents for self-evaluation and refinement. In this work, we leverage the agent verification module to verify its process in extrinsic early exit approach, and introduce two efficiency metrics to complement existing agent evaluation strategies.

8 Conclusion

In this work, We propose a dynamic early-exit framework for LLM-based agents in complex embodied environments, incorporating intrinsic and extrinsic early-exit mechanisms. Both approaches improve efficiency in our experiments. To better evaluate the impact of early exits, we introduce two complementary metrics that capture both its positive and negative effects. Additionally, we design a practical experiment in which a stronger agent assists a weaker one in continuing task execution, leading to enhanced performance. We hope our approach serves as a first step toward improving the efficiency of LLM-based agents and that our proposed metrics can be readily adopted by future research for evaluating agent efficiency.

Limitations

The limitations of our work are as follows:

- **Limited Datasets:** We evaluate only five datasets from embodied and gaming environments. Tasks like web navigation or app execution are excluded, as they often involve simpler, more direct goals, making early-exit less impactful. We leave these for future work.
- **No Training Integration:** While our approaches and metrics are designed to be plug-and-play for all LLM-based agents, we restrict our experiments to models that were not trained with held-in data due to uncertainties about the complexity of datasets.
- **LLM Scope:** We test four open-source LLMs due to budget constraints and to avoid data contamination. Proprietary models like GPT (Achiam et al., 2023) are not included.
- **Residual Redundancy:** While our approach reduces redundant steps, it does not fully eliminate them, likely due to current LLMs' limited instruction-following ability. Further improvements are still necessary.

Ethics Statement

We take ethical considerations very seriously, and strictly adhere to the Code of Ethics. All procedures performed in this study are in accordance with the ethical standards. This paper explores early-exit mechanisms for LLM-based agents in embodied environments. Our proposed approaches and metrics, does not include statements that induce the model to generate harmful information. Additionally, the approach focuses solely on determining when to terminate agent execution, thereby reducing potential risks. Both the datasets and models used in this paper are publicly available and have been widely adopted by researchers. We ensure that the findings and conclusions of this paper are reported accurately and objectively. No human participants were involved as evaluators or case studies in this work.

References

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

Ma Chang, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. 2024. Agentboard: An analytical evaluation board of multi-turn llm agents. *NeurIPS*.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. 2024a. Do not think that much for 2+3=? on the overthinking of o1-like llms. *arXiv preprint*.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. 2024b. Agent-FLAN: Designing data and methods of effective agent tuning for large language models. In *ACL*.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. 2019. Babyai: A platform to study the sample efficiency of grounded language learning. In *ICLR*.

Dayuan Fu, Keqing He, Yejie Wang, Wentao Hong, Zhuoma GongQue, Weihao Zeng, Wei Wang, Jingang Wang, Xunliang Cai, and Weiran Xu. 2025a. Agentrefine: Enhancing agent generalization through refinement tuning. In *ICLR*.

Dayuan Fu, Jianzhao Huang, Siyuan Lu, Guanting Dong, Yejie Wang, Keqing He, and Weiran Xu. 2025b. PreAct: Prediction enhances agent's planning ability. In *COLING*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint*.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2020. Interactive fiction games: A colossal adventure. In *AAAI*.

Sihao Hu, Tiansheng Huang, Gaowen Liu, Ramana Rao Kompella, Fatih Ilhan, Selim Furkan Tekin, Yichang Xu, Zachary Yahn, and Ling Liu. 2024. A survey on large language model-based game agents. *arXiv preprint*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *SOSP*.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024.

Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *MLSys*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2024. Agentbench: Evaluating llms as agents. In *The Twelfth International Conference on Learning Representations*.

Qingyu Lu, Baopu Qiu, Liang Ding, Kanjian Zhang, Tom Kocmi, and Dacheng Tao. 2024. Error analysis prompting enables human-like translation evaluation in large language models. In *ACL*.

Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, et al. 2025. Large language model agent: A survey on methodology, applications and challenges. *arXiv preprint*.

Siqi Ouyang and Lei Li. 2023. AutoPlan: Automatic planning of interactive decision-making tasks with large language models. In *EMNLP*.

Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. 2024. Autonomous evaluation and refinement of digital agents. In *COLM*.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *UIST*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *NeurIPS*.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. Alfworld: Aligning text and embodied environments for interactive learning. In *ICLR*.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024. Trial and error: Exploration-based trajectory optimization of LLM agents. In *ACL*.

Tianxiang Sun, Xiangyang Liu, Wei Zhu, Zhichao Geng, Lingling Wu, Yilong He, Yuan Ni, Guotong Xie, Xuanjing Huang, and Xipeng Qiu. 2022. A simple hash-based early exiting approach for language understanding and generation. In *ACL*.

Mauro Vallati, Lukas Chrpa, Marek Grześ, Thomas Leo McCluskey, Mark Roberts, Scott Sanner, et al. 2015. The 2014 international planning competition: Progress and trends. *Ai Magazine*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. ScienceWorld: Is your agent smarter than a 5th grader? In *EMNLP*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS*.

Yiran Wu, Tianwei Yue, Shaokun Zhang, Chi Wang, and Qingyun Wu. 2024. Stateflow: Enhancing llm task-solving through state-driven workflows. In *COLM*.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Zheng Lin, Li Cao, and Weiping Wang. 2025. Dynamic early exit in reasoning models. *arXiv preprint*.

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

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. In *AAAI*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *NeurIPS*.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. Bert loses patience: Fast and robust inference with early exit. *NeurIPS*.

A Recommended Early-Exit Approaches

Based on our experimental results and analysis, we provide a set of recommendations for selecting suitable early-exit approaches for specific LLMs. These guidelines are summarized in Table 3 and can serve as a reference for future research.

LLM	Intrinsic	Extrinsic
Llama3.1-8B-Instruct	◐	◐
Llama3.1-70B-Instruct	◐	◐
Mistral-7B-Instruct	◐	○
Mistral-24B-Instruct	○	◐

Table 3: **Recommendations for selecting early-exit approaches** for different LLMs.

B Prompt Variants

In our initial experiments, we observed that prompts behave differently across various LLMs. For instance, in the case of extrinsic early-exit, *Llama3.1-70B-Instruct* is particularly sensitive—strict prompts can easily trigger an early exit.

Early-Exit Approach	Strict Condition	Modest Condition
Intrinsic Early-Exit	Once the environment appears complete or no further progress is likely, include 'EXIT' in your action to end the task without delay.	If you believe the environment is complete, your task is finished, and no further attempts are needed, please include 'EXIT' in your action.
Prompt→LLM	*Llama3.1-70B-Instruct* *Mistral-24B-Instruct*	*Llama3.1-8B-Instruct* *Mistral-7B-Instruct*
Extrinsic Early-Exit	Evaluate the current history of the agent and determine if it meets any of the following conditions: 1. The recent steps show repetitive actions or the agent appears to be stuck in a loop. 2. The agent repeatedly checks for valid actions but fails to make meaningful progress toward the objective. 3. The agent's recent thoughts suggest the task is complete and no further steps are necessary. 4. The task is no longer achievable due to high difficulty or significant deviation from the expected course. If any of the above conditions are met, output "YES". Otherwise, output "NO" to indicate the agent should continue exploring.	Evaluate the agent's recent history and consider: 1. Whether the agent appears stuck or making little meaningful progress despite repeated attempts. 2. Whether the task seems complete or no longer feasible to pursue. If you have good reason to believe further steps are unlikely to help, you may output "YES" to suggest stopping. Otherwise, output "NO" and continue exploring.
Prompt→LLM	*Llama3.1-8B-Instruct* *Mistral-7B-Instruct* *Mistral-24B-Instruct*	*Llama3.1-70B-Instruct*

Table 4: Early-Exit prompt context with different condition. We also provide their correspondding LLM used in our approach.

To address this, we designed two prompt variants for each experimental setting: "Modest Condition" and "Strict Condition." The Strict Condition uses a firmer tone and outlines more detailed exit criteria, while the Modest Condition is more lenient. We provide the full prompt contexts in Table 4, along with their corresponding compatible LLMs.

C Prompt Context

We follow Chang et al. (2024) in using the provided task instruction, task goal, and example for each dataset. Since Chang et al. (2024) adopt an act-only prompting style rather than ReAct-style, we follow Song et al. (2024) to design a ReAct-style prompt format. The original examples are extended from Act-Only to ReAct-style using `gpt-4o-2024-08-06`. Initial observations and interactions are provided by the environment, and the intrinsic and extrinsic early-exit instructions are shown in Table 4. For ALFWorld and ScienceWorld tasks, we observe that providing valid actions leads to a significant performance difference (approximately 10%–20% in success rate). Therefore, we include valid actions in these two datasets to ensure fair comparison with prior work (Song et al., 2024; Fu et al., 2025a).

ReAct-Style Prompt for ALFWorld

SYSTEM:
You are a helpful assistant.
USER:
Your task is to interact with a virtual household simulator to accomplish a specific task. With each interaction, you will receive an observation. Your role is to ... {task instruction}
ASSISTANT: OK.
USER:
Here is the example:
{example}
Now, it's your turn. You should perform thoughts and actions to accomplish the goal. Your response should use the following format:
Thought: <your thoughts>
Action: <your next action>

Your task is: {task goal}
You are in the middle of a room. Looking quickly around you, ... {init observation}
{interaction history}

Important ##: Your thought should be short, clear and concise.
{intrinsic early-exit instruction}

The next action could be chosen from these valid actions: {valid actions}

Extrinsic Early-Exit Verification

SYSTEM:
You are a helpful assistant.
USER:
You will be given a historical scenario in which you are placed in a specific environment with a designated objective to accomplish.
Task Description: Your task is to interact with a virtual household simulator to accomplish a specific task. With each interaction, you will receive an observation. Your role is to ... {task instruction}
Your Objective:
{task goal}
Your Current History:
{interaction history}
Instructions:
{extrinsic early-exit instruction}
Do not include any additional text or explanations in your response.