Contrastive In-Context Learning with Active Memory for Task Planning

Jiho Lee[1]**, Kyungjae Lee**[2]**, and Eunwoo Kim**[1,*]
[1]School of Computer Science and Engineering, Chung-Ang University
[2]Department of Statistics, Korea University
j2hoooo@cau.ac.kr, kyungjae_lee@korea.ac.kr, eunwoo@cau.ac.kr

Abstract: Large language models (LLMs) have shown great promise in robotic task planning through in-context learning with a few successful demonstrations, guiding the model to generate feasible task plans. However, existing approaches typically overlook the learning signals from failures, often relying solely on manually curated positive examples. Recent methods attempt to utilize failures by transforming them into feedback or improved plans through LLMs, which involves additional steps to generate knowledge that can serve as positive examples for learning. Moreover, these methods often store examples indiscriminately in an external memory, leading to unbounded memory expansion and inefficiencies in both retrieval and resource usage. In this work, we propose a contrastive in-context learning framework for task planning that utilizes both successful and failed demonstrations through a dual prompting strategy. It enables the model to learn by contrasting positive prompts from successful ones, which guide correct behaviors, and negative prompts from failed ones, which prevent recurring mistakes. To implement this strategy efficiently, we introduce an active memory with a limited budget that selectively incorporates useful demonstrations generated by an LLM during task planning. Experiments on diverse multi-object, long-horizon, and spatially constrained manipulation tasks show that our method improves the average task success rate by 10.7%, while reducing memory size by up to 75.0% compared to existing LLM-based task planning approaches.

Keywords: Task planning, language model, contrastive in-context learning

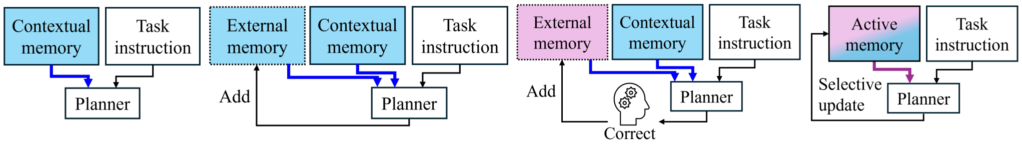
1 Introduction

Large language models (LLMs) have shown impressive generalization and reasoning capabilities across a wide range of tasks, from natural language understanding [1] to code generation [2], by leveraging extensive pre-trained knowledge. Recently, these capabilities have been extended to robotic task planning [3, 4, 5, 6, 7, 8, 9, 10, 11, 12], where an LLM generates action sequences through in-context learning [13]. In this approach, the model is provided with a few input-output examples (referred to as contextual memory) and generates plans without requiring additional fine-tuning. It enables task planning in scenarios with varying task specifications [3, 4, 5].

Most existing works employ contextual memory for in-context learning, typically composed of successful demonstrations manually curated by human experts [3, 4, 5, 6, 14], as illustrated in Figure 1(a). These approaches primarily focus on positive examples, which guide the model toward ideal action sequences. However, they overlook valuable learning signals from *negative examples*, which can help the model recognize and avoid incorrect behaviors [15].

In task planning, it is challenging to generate demonstrations that support learning from positive and negative examples. This is because such demonstrations require complex reasoning over temporal and causal dependencies, typically in a structured language-based representations, such as

[*] Corresponding author.



(a) Standard in-context learning

(b) In-context learning with external success memory

(c) In-context learning with external failure memory

(d) Ours: Contrastive in-context learning

Figure 1: **Comparison of in-context learning methods for task planning.** In memory, blue and purple colors represent examples derived from successful and failed demonstrations, respectively. Memories with dashed and solid borders indicate growing and fixed sizes, respectively. (a) learns with positive examples from a pre-defined contextual memory. (b) conducts in-context learning using successful demonstrations stored in an external memory, and (c) learns from corrected examples derived from failures in an external memory. In contrast, ours in (d) performs contrastive in-context learning with both successful and failure demonstrations by selectively incorporating them into an active memory of a limited size during task planning.

symbolic actions [6, 14] or planning domain languages [16, 17]. This leads to error-prone human-crafted examples, especially in long-horizon tasks with increased planning complexity. To address this, we utilize demonstrations generated by LLMs during task planning, where the LLM outputs the structured action sequences. Recently, a few works have leveraged generated demonstrations by storing them in an external memory for future knowledge retrievals when executing new task instructions [18, 19, 20]. They either collect only successful demonstrations [18] or corrected demonstrations from failures [19, 20], both of which also provide the model with positive examples, as in Figure 1(b) and 1(c). However, they lead to unbounded memory expansion and inefficient retrieval due to an extensive number of generated examples in the memory. Moreover, they often require an additional step to store or transform specific types of knowledge, such as for LLM querying.

To address these problems, we propose a novel approach that introduces *contrastive in-context learning* for task planning, utilizing both successful and failure demos through a dual prompting strategy. This method employs two prompts to learn by exploiting the contrast between them, where positive prompts derived from successes guide desirable reasoning patterns, and negative prompts from failures help in recognizing incorrect behaviors. To make use of both types of demos, we introduce an active memory that serves as a self-evolving repository of LLM-generated demos to provide task-specific knowledge while selectively integrating useful ones. Additionally, we adopt a diversity-aware selection strategy to ensure a compact yet informative memory, identifying a representative set of demos. In doing so, the robot can continually improve its planning ability by distilling its own experiences, including successes and failures, without requiring additional supervision or resources.

To validate the effectiveness of our method, we conduct evaluations on a diverse set of benchmark scenarios, each reflecting key challenges in LLM-based robot task planning: Shelf [21] for multi-object manipulation, Coffee [5] for long-horizon task sequences, and Unpacking [22] for spatially constrained tasks. Experimental results demonstrate that the proposed contrastive in-context learning approach significantly outperforms existing in-context learning based task planning methods, improving task success rates by up to 20.8% and reducing memory usage by up to 75.0%. Furthermore, we demonstrate the real-world applicability of our approach, showing that leveraging both successful and failed cases through contrastive in-context learning helps prevent potential mistakes, compared to the standard in-context learning, which relies on pre-defined successful examples.

2 Related Works

To leverage pre-trained large language models (LLMs) without additional parameter updates, in-context learning [13] has emerged as a powerful learning strategy. LLMs can rapidly adapt to new tasks and generate contextually appropriate responses based on a small set of input-output examples. This capability has recently been applied to robotic task planning [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 18, 19, 20, 23, 24], where in-context examples facilitate the generation of coherent action sequences, showcasing its flexibility and generalization ability.

Most works focus on generating plans grounded in the environment [3, 4, 6, 7, 8, 9]. Specifically, [3] combines LLMs for task-grounding information with affordance functions to consider the environment information. [4] employs a scene descriptor with a visual question answering model [25] to obtain environment feedback. [6] converts natural language instructions into the planning domain definition language (PDDL) formulation [16, 17] with an LLM-based translator. [8] presents a partially observable task planning framework by considering uncertainties of the environment and skill execution. Additionally, some works generate an executable code that represents task plans [10, 11, 14]. [10, 14] present programmatic prompting templates to generate a structured robot-executable code. [11] develops multi-level task decomposition for a multi-robot framework in code-style representations. These methods rely on manually curated successful examples in a pre-defined contextual memory for in-context learning, limiting the ability of the model to learn from diverse examples.

To address this limitation, recent works have explored memory-augmented approaches that incorporate demonstrations generated by LLMs during task planning into an external memory [18, 19, 20], thereby enhancing the learning capability of the model. [18] collects successful experiences, and [19, 20] store transformed feedback or improved plans obtained by querying an LLM with failed outputs. However, these methods face several limitations. First, they rely on positive prompting, where the model is encouraged during training to imitate the provided positive examples. Second, they often suffer from unbounded memory growth, leading to resource overhead and inefficient retrieval. Third, they typically store only a subset of experiences either successful demonstrations [18] or corrections derived from failures [19, 20], which limits the diversity of learned knowledge. In contrast, we propose contrastive in-context learning with dual prompting, which provides both positive and negative examples to guide the model on what to imitate and what to avoid. To realize this in a resource-constrained robotic system, we introduce an active memory module that continuously curates and updates a compact collection of successful and failed LLM-generated demonstrations, selectively retaining the most useful examples within a limited memory budget.

Additionally, other recent studies have focused on detecting and correcting errors in action sequences generated by LLMs [5, 12, 23, 24, 26]. [24] employs an external validator with a rule-based verifier, while [5] uses additional LLMs to summarize robot sensory observations for failure reasoning. [23, 26] propose an iterative self-refinement process, and [12] introduces an execution-level feedback loop with a vision-language model. In contrast, our approach enhances planning capabilities through the proposed contrastive in-context learning, without relying on corrected examples or requiring additional re-planning processes.

3 Method

3.1 Overview

Our goal is to enable large language models (LLMs) to enhance their learning capabilities by fully utilizing both self-generated successful and failed experiences under a limited memory budget. Typically, existing works [3, 4, 10, 14] perform standard in-context learning using a fixed contextual memory, M_F, manually constructed by human experts, as illustrated in Figure 2(a). $M_F = \{d_1, \ldots, d_B\}$ is a set of B successful demonstrations, where each $d_i = (x_i, y_i)$ consists of an input-output pair that encourages the model to imitate the provided behaviors as positive prompting. To broaden the learning capability using self-generated new experiences during task planning, recent works have attempted to store them in an external memory by transforming and accumulating them [18, 19, 20], as depicted in Figure 2(b) (transforming can be omitted). However, they indiscriminately accumulate data into an unbounded memory, hindering efficient retrieval, and often require additional steps to store or transform specific types of knowledge for positive prompting [19, 20].

To address these limitations, we introduce a contrastive in-context learning framework with an active memory to effectively leverage both generated successful and failed knowledge during task planning. The framework employs a dual prompting strategy that contrasts positive prompts from successes and negative prompts from failures to allow the model to imitate correct behaviors and avoid mistakes. As shown in Figure 2(c), the planner generates an action sequence as a demon-

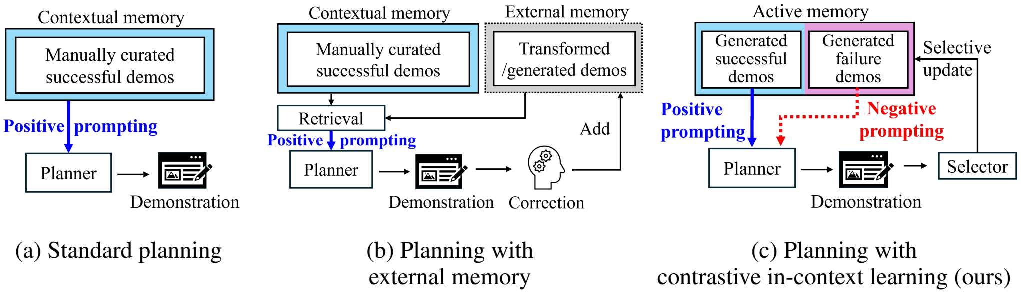


Figure 2: **A comparison of approaches for LLM-based task planning.** (a) generates a demonstration as an action sequence by performing positive prompting with manually curated successful demonstrations to guide the planner toward optimal planning behaviors. (b) conducts positive prompting using an external memory that retains successful demonstrations or refined demonstrations obtained from the correction of failures. (c) employs positive and negative prompting by utilizing generated successful and failed demonstrations stored in an active memory, respectively.

stration for the given task instruction, resulting in success or failure. A selector module evaluates existing and newly generated demonstrations by estimating their utility using a diversity-aware approach in the language feature space. It updates the active memory by selecting the most informative subset while preserving the fixed capacity of M_F. The updated active memory is then used to provide contrastive examples during in-context learning, enabling an LLM to continuously refine its planning ability with a compact yet diverse set of experiences.

3.2 Active Memory

To enable experience-driven learning within resource-constrained robotic systems, it is crucial to build a memory as a diverse set of demonstrations while minimizing human efforts and additional resources. To this end, we construct an active memory, M_A, consisting of demonstrations generated by an LLM, preserving its planning-oriented language and reasoning patterns. Notably, we retain both successful and failed demonstrations in M_A, as each provides complementary guidance in learning to capture the reasoning patterns behind success or failure.

The active memory is progressively updated by a selector module, $\mathcal{S}$, which employs a diversity-aware selection strategy inspired by active learning [27]. It evaluates both newly generated and stored demonstrations to identify the most informative and representative subset within a constrained memory budget. Given the current memory, M_A, and a set of newly generated demonstrations, G, containing successful and failed cases, the selector evaluates them in the language feature space to update the memory by discarding less informative or redundant examples and incorporating those that enhance diversity and representativeness in memory. This can be formalized as follows:

$$M_A \leftarrow \mathcal{S}(M_A, G), \text{ where } M_A = \{M_A^s, M_A^f\}. \tag{1}$$

Here, M_A^s and M_A^f represent the sets of selected successful and failed demonstrations, respectively.

To identify the most valuable demos, we first encode them into vector representations using Sentence-BERT [28], capturing their semantic and contextual features. Then, we perform k-means clustering [29] on successful and failed demos separately, aiming to ensure diversity within each case. The total number of clusters is set to the memory budget, and the allocation between success and failure clusters can be adjusted, whose impact is analyzed in Section 4.1.2. From each cluster, we select a representative demo to retain in the active memory, keeping it compact yet informative.

3.3 Contrastive In-Context Learning with Dual Prompting

We introduce contrastive in-context learning with dual prompting to enhance in-context learning while adaptively leveraging two types of demonstrations. The key intuition is that effective learning benefits not only from understanding correct reasoning patterns but also from learning to recognize

Positive prompting (g_p, x_i, y_i)

This is a *success and correct* plan. *Learn* this example and *follow* the pattern and logic demonstrated.

- Initial state: {`(empty cup1)`}
- Goal state: {`(fill cup1 coffee)`}
- Plan: {`(pick cup1 table)` `(place cup1 coffee-machine)` `(toggle-on coffee-machine)` `(toggle-off coffee-machine)` `(pick cup1 coffee-machine)` `(place cup1 table)`}

Negative prompting (g_n, x_i, y_i)

This is a *failure and incorrect* plan. *Do not learn* this example and *avoid* the pattern and logic shown.

- Initial state: {`(empty cup1)`}
- Goal state: {`(fill cup1 coffee)`}
- Plan: {`(pick cup1 table)` `(place cup1 coffee-machine)` `(toggle-off coffee-machine)` `(pick cup1 coffee-machine)` `(place cup1 table)`}

Figure 3: **Examples of positive and negative prompting.** (a) Positive prompting helps the model learn correct behaviors through an instruction and a successful demonstration, while (b) negative prompting encourages avoiding incorrect behaviors with a negative learning signal and a failed demonstration. Underlined texts indicate the differences between the two demonstrations.

and avoid unsuccessful ones. This is motivated by the observation that LLMs are highly sensitive to the context and can imitate behaviors depending on the demonstrations they are given [30]. It can address the limitation of conventional prompting strategies, which primarily focus on positive examples [13], overlooking valuable learning signals from failure cases.

Accordingly, the active memory with successful demonstrations M_A^s and failure ones M_A^f is utilized for positive and negative prompting, respectively. Positive prompting provides successful exemplars that demonstrate how to solve a task [13], whereas negative prompting introduces failure cases to help the model recognize and avoid incorrect strategies, inspired by [31]. Each demonstration $d_i = (x_i, y_i)$ is reconstructed into a triplet (g, x_i, y_i), where g is an instruction signal determined by the prompting type as follows:

$$(g, x_i, y_i) = \begin{cases} (g_p, x_i, y_i), & \text{if } (x_i, y_i) \in M_A^s, \\ (g_n, x_i, y_i), & \text{if } (x_i, y_i) \in M_A^f. \end{cases} \tag{2}$$

Here, g_p is a positive instruction that encourages the model to imitate successful behavior, while g_n is a negative instruction that discourages the replication of failure. These instructions implicitly label the demonstrations, enabling dual prompting to contrast successes and failures within the context. Examples of positive and negative prompting are shown in Figure 3.

4 Experiments

4.1 Benchmark Experiments

4.1.1 Setup

Datasets. We validated the proposed method in three widely-used benchmark scenarios, including Shelf [21], Coffee [5], and Unpacking [22]. Each scenario is designed to validate the capability of the LLM-based task planning in multi-object, long-horizon, and spatially constrained manipulation tasks, respectively. In the Shelf scenario, the robot aims to re-arrange N objects, initially placed at random, into a size-ordered sequence on a shelf that can hold up to $N+1$ objects. The objective of the Coffee scenario is to serve N cups of coffee, each composed of up to six different ingredients, using one coffee machine. The Unpacking scenario requires retrieving a designated target object from a box filled with N objects of varying sizes and placing it into a separate box. Each environment was evaluated using 20 randomized test cases, designed with different initial and goal configurations. To evaluate performance across tasks of varying complexities, we adjusted the number of objects N in each scenario. Specifically, we set N to 2 and 4 for Coffee, and 5 and 7 for both Shelf and Unpacking. This variation produces task sequences of different lengths. In the Shelf scenario, the average sequence spans 15 and 26 steps for $N = 5$ and $N = 7$, respectively. In the Coffee scenario, the averages increase to 29 and 57 for $N = 2$ and $N = 4$, respectively. The Unpacking scenario has 5 and 7 steps for $N = 5$ and $N = 7$.

The task sequences are relatively short since accessing the target involves removing only a few obstructing objects in the box, not all items.

Table 1: **Comparison results of task success rates.** The bold and underlined texts indicate the best and second-best results, respectively.

Method	Shelf		Coffee		Unpacking		Avg.
	$N = 5$	$N = 7$	$N = 2$	$N = 4$	$N = 5$	$N = 7$	
GPT-4o-mini							
Standard planning	72.5%	35.0%	77.5%	62.5%	62.5%	32.5%	57.1%
Re-planning	90.0%	37.5%	85.0%	77.5%	67.5%	37.5%	65.8%
Planning w/ external success memory	95.0%	50.0%	**90.0%**	85.0%	77.5%	45.0%	73.8%
Planning w/ external failure memory	90.0%	47.5%	87.5%	82.5%	77.5%	52.5%	72.9%
Planning w/ active memory (ours)	**97.5%**	**52.5%**	**90.0%**	**87.5%**	**82.5%**	**57.5%**	**77.9%**
Gemini-1.5-Flash-8B							
Standard planning	52.5%	20.0%	70.0%	57.5%	35.0%	25.0%	43.3%
Re-planning	55.0%	20.0%	85.0%	55.0%	37.5%	22.5%	45.8%
Planning w/ external success memory	57.5%	**30.0%**	**90.0%**	**80.0%**	47.5%	25.0%	55.0%
Planning w/ external failure memory	65.0%	22.5%	85.0%	67.5%	52.5%	20.0%	52.1%
Planning w/ active memory (ours)	**67.5%**	27.5%	**90.0%**	**80.0%**	**60.0%**	**35.0%**	**60.0%**

Implementation details. To verify the applicability of the proposed method across diverse models, we used two models: GPT-4o-mini (GPT) [32] and Gemini-1.5-Flash-8B (Gemini) [33] both of which have approximately 8 billion parameters. We utilized the LLM to convert natural language task instructions into PDDL representations [6], with a temperature of 0 to eliminate randomness in the output. After that, the LLM generated action sequences based on the translated PDDL, using a temperature of 0.1 to allow for diversity in plan generation. The success or failure of the generated plan was assessed using a rule-based simulation system [23]. Additionally, we set the memory budget (B) to 6 and 12 demonstrations, updating the active memory every five task instructions to balance learning responsiveness and stability. To balance the ratios of successful and failed examples in memory, we limited the number of failed examples to at most half of the memory budget (see Table 3 for analyses of the failure ratio).

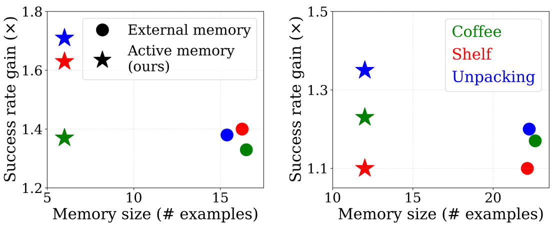
Comparison methods. We compared the proposed method against four representative baselines. (i) *Standard planning* performs in-context learning for task planning by using a fixed set of successful demonstrations with positive prompting [3, 4, 6]. (ii) *Re-planning* corrects errors of the generated action sequence through positive prompting with resolved correction examples as a recent self-corrective method [23]. (iii) *Planning with external success memory* [18] stores demonstrations that are either initially successful or become successful after re-planning and continues positive prompting with them. (iv) *Planning with external failure memory* [19, 20], stores the original failed demonstrations and their improved ones obtained through re-planning, along with feedback. For (iii) and (iv), we apply a recent self-corrective approach [23] for re-planning and use the widely adopted Sentence-BERT [28] to retrieve demonstrations with the closest embeddings to the task instruction.

4.1.2 Results

Main results. We summarize the comparison results on three benchmark scenarios in Table 1, averaging over two memory budgets. The proposed method consistently outperforms all competitors, achieving an average increase of 10.5% with GPT and 11.9% with Gemini, respectively. Compared to Standard planning, our approach demonstrates a 18.8% higher average success rate across all scenarios. This indicates that relying on a pre-defined contextual memory limits the learning capability of the model and its adaptability in diverse scenarios. Although Re-planning requires additional resources, such as querying an LLM for plan validation and generating feedback, it shows lower average performance than our method with a 13.2% drop across all settings. External memory with successful examples yields comparable results to our proposal in some scenarios, but it shows an average performance drop of about 10.1% in the most challenging Unpacking scenario. We observe that external failure memory method consisting of corrected failures achieves performance similar to using successful examples, but it still results in an 6.5% average drop compared to ours across all scenarios. Those results suggest that incorporating both successful and failed demonstrations enhances the learning capabilities of LLMs by explicitly indicating which plans to imitate and which to avoid, thereby compensating for the limited guidance implicitly offered in successful examples.

Table 2: **Ablation study in terms of active memory, dual prompting, and selector.**

Functionality			Shelf		Coffee		Unpacking		Avg.
Active memory	Dual prompting	Selector	$N = 5$	$N = 7$	$N = 2$	$N = 4$	$N = 5$	$N = 7$	
GPT-4o-mini									
			72.5%	35.0%	77.5%	62.5%	62.5%	32.5%	57.1%
✓			82.5%	40.0%	85.0%	75.0%	72.5%	45.0%	66.7%
✓		✓	92.5%	47.5%	87.5%	80.0%	72.5%	50.0%	71.7%
✓	✓		90.0%	45.0%	87.5%	82.5%	75.0%	50.0%	71.7%
✓	✓	✓	**97.5%**	**52.5%**	**90.0%**	**87.5%**	**82.5%**	**57.5%**	**77.9%**
Gemini-1.5-Flash-8B									
			52.5%	20.0%	70.0%	57.5	35.0%	25.0%	43.3%
✓			60.0%	22.5%	80.0%	72.5%	45.0%	27.5%	51.3%
✓		✓	62.5%	25.0%	80.0%	75.0%	47.5%	27.5%	52.9%
✓	✓		65.0%	25.0%	85.0%	75.0%	52.5%	32.5%	55.8%
✓	✓	✓	**67.5%**	**27.5%**	**90.0%**	**80.0%**	**60.0%**	**35.0%**	**60.0%**



(a) $B = 6$

(b) $B = 12$

Figure 4: **Comparison of success rate and memory size.** Success rate gain denotes the success rate improvement compared to Standard planning.

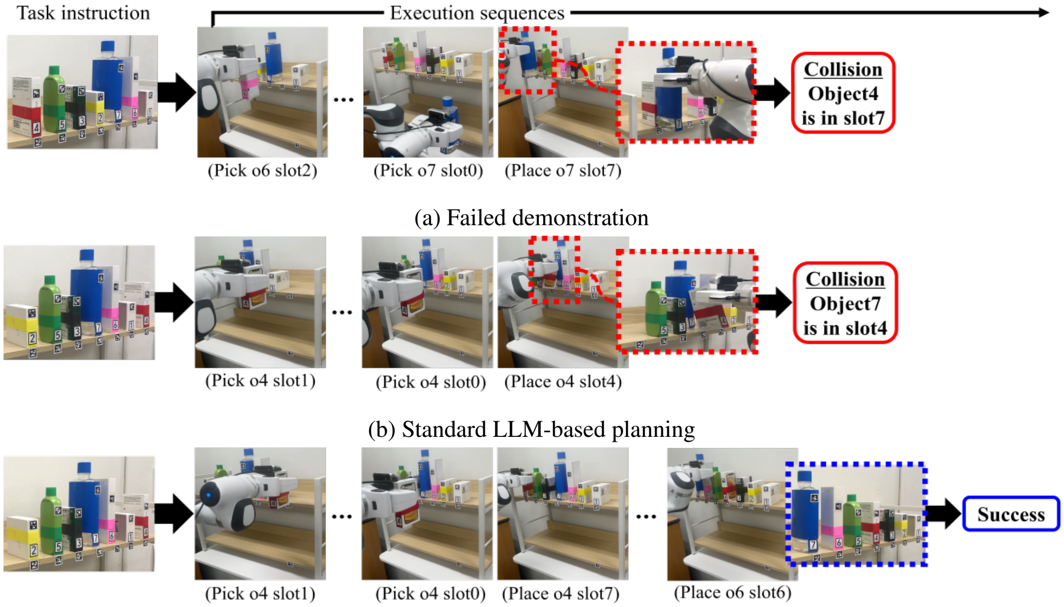
Table 3: **Analysis of prompt ratio.** The prompt ratio denotes the maximum proportion of negative prompts within the active memory. The values indicate task success rate gains over the 0% prompt ratio baseline.

Prompt ratio	0%	33%	50%	67%
$B = 6$				
Easy	-	+4.2%	**+11.7%**	+5.8%
Hard	-	+5.0%	**+8.3%**	+1.7%
$B = 12$				
Easy	-	+0.8%	**+4.2%**	+1.7%
Hard	-	+2.5%	**+3.3%**	+1.7%

Ablation study. We conducted an ablation study to investigate the impact of the proposed method including the active memory, dual prompting, and the selector. The results are summarized in Table 2, averaged across two budgets. Employing only the active memory yields a modest 8.8% improvement over not employing it across all scenarios. However, it still shows an average 11.2% drop compared to ours. Interestingly, comparing the impact of the selector and dual prompting, we observe that the absence of each resulted in an average drop of 5.2% and 6.7%, respectively, across all settings. It indicates that dual prompting, which leverages successful and failed knowledge for contrastive in-context learning, improves the overall performance and adaptability of the model.

Efficiency of active memory. We evaluated our active memory and the external memory approach [18, 19, 20], comparing average success rate gain over standard planning and memory size, as shown in Figure 4. External memory methods expand as new demos accumulate over time, thereby requiring larger memory resources, averaging about 2.27 times the size of the proposed active memory across all scenarios. They achieve a 1.26 average success rate gain over standard planning, whereas our proposal achieves an higher average gain of 1.38, demonstrating superior performance with substantially less memory. These findings indicate that integrating newly generated knowledge while optimizing memory allocation can cut memory usage without compromising performance.

Impact of prompt ratio. We analyzed the impact of varying the ratio of negative prompts given an active memory budget (B) of 6 and 12, as shown in Table 3. We evaluated four configurations using GPT and Gemini: 0% with positive prompts only and allowing up to 33%, 50%, and 67% negative prompts out of each memory budgets, respectively. For the analysis, we grouped scenarios into two categories, with settings for each scenario with fewer objects considered as Easy and settings with more objects considered as Hard. Then, we calculated the average success rate for each category and reported the improvements over the method using only positive prompts (0%). On average, the results show that maintaining a balanced ratio of positive and negative signals yields the best performance. Notably, increasing the proportion of negative prompts tends to benefit Easy, whereas increasing the proportion of positive prompts is more effective for Hard.



(a) Failed demonstration

(b) Standard LLM-based planning

(c) Planning with contrastive in-context learning (ours)

Figure 5: **Comparison of execution sequences in the Shelf scenario between the standard LLM-based planning and the proposed planning method.** (`action` oi `slot`j) represents that the `action` targets `object`i (denoted by oi), taking `slot`j into account. Best viewed in color (×2).

4.2 Real-world Experiments

We evaluated the proposed method using a 7-DOF Franka Emika Panda robot arm in the Shelf scenario using AprilTag [34] for scene perception with a wrist camera. The robot rearranges seven objects of different sizes on a shelf according to a specified size-based order using Gemini. Note that the shelf has one extra slot available. To gather scene information for all objects and slots, the robot returns to detecting positions after each action. We compared the generated action sequences against the Standard planning approach. This allows us to evaluate both the planning performance and the impact of the proposed contrastive in-context learning with successes and failures.

As shown in Figure 5(a), both the standard and proposed planning approaches fail when executing (`place` o7 `slot7`), since `slot7` is already occupied by `object4`, resulting in a collision. Then, the standard planning method fails again given a similar task instruction, as illustrated in Figure 5(b). Since it does not leverage the previous failed experience for learning, resulting in a similar collision when attempting (`place` o4 `slot4`), where `slot4` is already occupied by `object7`. However, ours leverages the previous failed experience through contrastive in-context learning, allowing the model to learn to avoid similar mistakes and complete a similar task in the future, as shown in Figure 5c.

5 Conclusion

In this work, we have introduced a contrastive in-context learning framework for task planning with large language models. It employs a dual prompting strategy, where successful knowledge serves as positive prompts to guide the model toward effective behaviors, and failed knowledge is used as negative prompts to help avoid previously observed mistakes. It allows the model to learn from the contrast between the two different prompts. To leverage both types of knowledge for learning, we introduce an active memory that is progressively updated with task-specific demonstrations, generated by LLMs during task planning, ensuring the retention of a compact but diverse set of knowledge. Extensive evaluations in both benchmark environments and real-world settings shows that our method achieves a markedly higher task completion rate than existing LLM-based planners while requiring only minimal additional resources.

Acknowledgment. This work was supported by the Institute of Information Communications Technology Planning Evaluation (IITP) grant funded by the Korea government (MSIT) [RS-2021-II211341, Artificial Intelligence Graduate School Program (Chung-Ang University)].

References

[1] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[2] E. Nijkamp, B. Pang, H. Hayashi, L. Tu, H. Wang, Y. Zhou, S. Savarese, and C. Xiong. Codegen: An open large language model for code with multi-turn program synthesis. In *The Eleventh International Conference on Learning Representations*, 2023.

[3] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on robot learning*, pages 287–318. PMLR, 2023.

[4] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. In *Conference on Robot Learning*, pages 1769–1782. PMLR, 2023.

[5] Z. Liu, A. Bahety, and S. Song. Reflect: Summarizing robot experiences for failure explanation and correction. In *Conference on Robot Learning*, pages 3468–3484. PMLR, 2023.

[6] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023.

[7] J. Shin, J. Han, S. Kim, Y. Oh, and E. Kim. Task planning for long-horizon cooking tasks based on large language models. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13613–13619. IEEE, 2024.

[8] L. Sun, D. K. Jha, C. Hori, S. Jain, R. Corcodel, X. Zhu, M. Tomizuka, and D. Romeres. Interactive planning using large language models for partially observable robotic tasks. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14054–14061. IEEE, 2024.

[9] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2998–3009, 2023.

[10] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE, 2023.

[11] S. S. Kannan, V. L. Venkatesh, and B.-C. Min. Smart-llm: Smart multi-agent robot task planning using large language models. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12140–12147. IEEE, 2024.

[12] Y. Guo, Y.-J. Wang, L. Zha, and J. Chen. Doremi: Grounding language model by detecting and recovering from plan-execution misalignment. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12124–12131. IEEE, 2024.

[13] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[14] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE, 2023.

[15] A. N. Meltzoff. Understanding the intentions of others: re-enactment of intended acts by 18-month-old children. *Developmental psychology*, 31(5):838, 1995.

[16] C. Aeronautiques, A. Howe, C. Knoblock, I. D. McDermott, A. Ram, M. Veloso, D. Weld, D. W. Sri, A. Barrett, D. Christianson, et al. Pddl—the planning domain definition language. *Technical Report, Tech. Rep.*, 1998.

[17] P. Haslum, N. Lipovetzky, D. Magazzeni, C. Muise, R. Brachman, F. Rossi, and P. Stone. *An introduction to the planning domain definition language*, volume 13. Springer, 2019.

[18] G. Sarch, Y. Wu, M. Tarr, and K. Fragkiadaki. Open-ended instructable embodied agents with memory-augmented large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3468–3500, 2023.

[19] L. Zha, Y. Cui, L.-H. Lin, M. Kwon, M. G. Arenas, A. Zeng, F. Xia, and D. Sadigh. Distilling and retrieving generalizable knowledge for robot manipulation via language corrections. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15172–15179. IEEE, 2024.

[20] L. Bärmann, R. Kartmann, F. Peller-Konrad, J. Niehues, A. Waibel, and T. Asfour. Incremental learning of humanoid robot behavior from natural interaction and large language models. *Frontiers in Robotics and AI*, 11:1455375, 2024.

[21] A. Krontiris and K. E. Bekris. Dealing with difficult instances of object rearrangement. In *Robotics: Science and Systems*, volume 1123, 2015.

[22] D. Jin, J. Park, and K. Lee. Perturbation-based best arm identification for efficient task planning with monte-carlo tree search. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5758–5764. IEEE, 2023.

[23] Z. Zhou, J. Song, K. Yao, Z. Shu, and L. Ma. Isr-llm: Iterative self-refined large language model for long-horizon sequential task planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2081–2088. IEEE, 2024.

[24] S. S. Raman, V. Cohen, I. Idrees, E. Rosen, R. Mooney, S. Tellex, and D. Paulius. Cape: Corrective actions from precondition errors using large language models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14070–14077. IEEE, 2024.

[25] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.

[26] J. Lee, H. Lee, J. Kim, K. Lee, and E. Kim. Self-corrective task planning by inverse prompting with large language models. *arXiv preprint arXiv:2503.07317*, 2025.

[27] Z. Zhang, E. Strubell, and E. Hovy. A survey of active learning for natural language processing. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6166–6190, 2022.

[28] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, 2019.

[29] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[30] Y.-M. Tseng, Y.-C. Huang, T.-Y. Hsiao, W.-L. Chen, C.-W. Huang, Y. Meng, and Y.-N. Chen. Two tales of persona in llms: A survey of role-playing and personalization. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 16612–16631, 2024.

[31] M. Pawelczyk, S. Neel, and H. Lakkaraju. In-context unlearning: language models as few-shot unlearners. In *Proceedings of the 41st International Conference on Machine Learning*, pages 40034–40050, 2024.

[32] OpenAI. Gpt-4o mini: advancing cost-efficient intelligence. 2024. URL `https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/`.

[33] Google. Gemini models, 2025. URL `https://ai.google.dev/gemini-api/docs/models/gemini`.

[34] E. Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE international conference on robotics and automation*, pages 3400–3407. IEEE, 2011.