# Amortizing Self-Correction for Continual Embodied Adaptation

**Woo Kyung Kim[1], Jooyoung Kim[1], Minjong Yoo[2]*, Honguk Woo[1]†**
[1]Department of Computer Science and Engineering, Sungkyunkwan University
[2]Samsung Electronics
kwk2696@skku.edu, onsaemiro@skku.edu
min-jong.yoo@samsumg.com, hwoo@skku.edu

## Abstract

Language model (LM)-powered agents have shown promise in addressing complex embodied tasks by leveraging the commonsense world knowledge of pre-trained models in interactive physical environments. Yet, real-world deployment requires agents to continually adapt to a stream of tasks, where neither large-scale expert supervision nor reliable external feedback is consistently available. In this paper, we present ASC, a framework for efficient continual task adaptation in embodied planning through amortized self-correction. Specifically, ASC trains the policy under two objectives: a standard CoT reasoning objective distilled from stronger experts, and a reflexive reasoning objective that internalizes self-correction by correcting initial rationales generated by a previously learned policy. This dual optimization allows the agent to adapt more efficiently through forward knowledge transfer over time, ultimately fostering robust CoT reasoning at inference without requiring explicit reflection. Furthermore, we introduce rationale-wise test-time scaling that dynamically adjusts the depth of CoT reasoning based on the model's confidence in actions inferred from its own rationales. Through experiments on VirtualHome and ALFWorld, we demonstrate performance superiority over other LM-based planning and continual task adaptation approaches, while achieving strong efficiency in computation, data usage.

## 1 Introduction

Recent advances in language model (LM)-powered agents have demonstrated strong capabilities in tackling complex, open-ended embodied tasks by grounding the commonsense world knowledge of pre-trained models into interactive physical environments (Ahn et al., 2022; Driess et al., 2023; Li et al., 2025). Despite these advancements, prior approaches rely either on large-scale supervision from expert models (e.g., CoT distillation) (Choi et al., 2024; DeepSeek-AI et al., 2025) or external signals to self-correct erroneous rationales (Welleck et al., 2023; Shinn et al., 2023).

In real-world applications, embodied agents are required to adapt to a stream of tasks over time, as everyday embodied tasks continually shift in surrounding objects, their relations, required skills, and environmental dynamics (Powers et al., 2022; Li et al., 2024; Kim et al., 2024). However, providing sufficient supervision for every new adaptation target becomes infeasible. Moreover, stronger experts are rarely accessible for external feedback in practical deployment, and even when such feedback is available, self-correction inherently requires two separate reasoning steps (i.e., one for initial reasoning and another to explicitly revise it upon error detection), doubling the inference cost.

To address these challenges, we propose to amortize self-correction across the continual stream of tasks, internalizing the corrective reasoning into the policy itself rather than invoking it explicitly at inference time. Specifically, at each task adaptation stage, the learning process involves dual supervision: a standard objective of learning CoT reasoning through distillation from stronger experts, and a complementary objective of learning reflexive reasoning through the correction of initial rationales generated by a previously learned policy. This reflexive reasoning provides a richer learning signal beyond mere CoT distillation by identifying and correcting discrepancies between earlier reasoning and new contexts – a form of metacognitive behavior that fosters transferable knowledge in evolving environments (Yeung and Summerfield, 2012). In this way, the dual supervision reinforces CoT reasoning via internalized reflexive knowledge during training, thereby delivering robust performance under few-shot expert supervision without requiring explicit reflection at inference.

*Work done while at Sungkyunkwan University.
†Corresponding author.



*Proceedings of the 15th International Joint Conference on Natural Language Processing and the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 5858–5878
November 6-10, 2026 ©2026 Association for Computational Linguistics

To this end, we present the ASC framework, designed for continual task adaptation in embodied planning by ***amortizing self-correction***. At each adaptation stage, we first retrieve the most relevant previously learned policy by comparing the rationales from previous stages with those derived from the current stage's demonstrations. The current policy is initialized with the retrieved prior one and then optimized using dual objectives: generating rationales via CoT reasoning, and correcting rationales generated by the prior policy via reflexive reasoning, both guided by few-shot demonstrations. Following this adaptation, inference is conducted solely through CoT reasoning, without explicitly invoking reflexive reasoning. To improve efficiency, we devise ***rationale-wise test-time scaling*** which dynamically determines when to terminate CoT reasoning based on the model's confidence in the actions predicted from the generated rationales.

We evaluate ASC on embodied planning benchmarks, which require generating high-level semantic plans from natural language instructions. Through evaluation on VirtualHome (Puig et al., 2018) and ALFWorld (Shridhar et al., 2021) benchmarks, we demonstrate the effectiveness of ASC over existing LM-based planning and continual task adaptation baselines across three aspects. (i) Continual few-shot adaptation performance: ASC achieves robust task success, with an average improvement of 19.11% over the most competitive baseline SeqFT-Distill (Shridhar et al., 2023) on the unseen category in the continual 5-shot adaptation setting (see Table 2). (ii) Computational efficiency: ASC substantially reduces the computation cost of CoT reasoning, lowering the number of generated tokens by 46.01% (see Table 5), while still outperforming the Self-Correction (Welleck et al., 2023), which requires nearly twice as many rationale generation for refinement, with an average improvement of 25.78% (see Table 3). (iii) Data efficiency: ASC demonstrates strong data efficiency, outperforming SeqFT-Distill trained with twice the demonstrations by 7.50% and TAIL-Distill (Liu et al., 2024) by 18.75% on the unseen category (see Table 6). The contributions of our work are summarized as follows.

- We present the ASC framework to address the challenges of efficient embodied task adaptation.
- We devise amortizing self-correction, a joint training strategy that combines CoT reasoning and reflexive reasoning, supervised respectively by few-shot CoT and rationale-wise correction of prior knowledge. Notably, this yields asymmetric effects, strengthening CoT reasoning at inference without requiring explicit reflection.
- We introduce rationale-wise test-time scaling, by which the depth of CoT reasoning is dynamically adjusted. This mechanism improves computational efficiency and enhances task success, thereby removing the need for explicit self-correction that would otherwise impose additional inference overhead.
- Through extensive experiments on VirtualHome and ALFWorld benchmarks, we demonstrate that ASC achieves superior task performance with improved data and computational efficiency.

## 2 Related Work

**Embodied task planning.** In recent years, embodied task planning has gained significant attention, driven by the advancements in LMs' reasoning capabilities (Driess et al., 2023; Huang et al., 2022a, 2023). Such progress enables agents to handle complex scenarios such as everyday household tasks by generating natural-language plans grounded in the environment. SayCan (Ahn et al., 2022) is one of the pioneering works leveraging LMs for embodied tasks, exploiting skill affordance functions to guide the generation of feasible actions. DeDer (Choi et al., 2024) distills the reasoning capabilities of LLMs into small LMs through a two-tier hierarchy. Another line of work integrates various sources of feedback to replan or correct decision-making at inference time (Huang et al., 2022b; Shinn et al., 2023; Oh et al., 2023). Unlike these approaches, which treat CoT reasoning and self-correction as separate inference-time stages, ASC unifies them by enhancing CoT reasoning itself through *reflexive reasoning*, enabling LM-based agents to efficiently adapt to new tasks under limited supervision and without explicit correction at inference.

**Continual task adaptation.** Prior works (Gao et al., 2021; Wan et al., 2024) have focused on leveraging a stream of datasets to progressively learn diverse tasks. TAIL (Liu et al., 2024) and L2M (Schmied et al., 2023) adopt parameter-efficient tuning methods to harness the knowledge embedded in pre-trained models for efficient adaptation in robotic manipulation. CAMA (Kim et al., 2024) proposes a framework for continual embodied planning, where model updates are guided by

previously stored logits to prevent catastrophic forgetting. Our work distinguishes itself by structuring robust CoT reasoning within LM-based policies through amortizing self-correction into the policy.

## 3 Preliminaries

### 3.1 Problem formulation

We consider a stream of few-shot demonstrations $\{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_H\}$, where $i$-th stage demonstrations $\mathcal{D}_i$ contain expert transitions. Each transition $d = (\mathcal{T}, o, a, \mathcal{Z})$ is represented as a tuple of task $\mathcal{T}$, observation $o$, action (plan) $a$, and a rationale set $\mathcal{Z}$. A task is defined by an underlying instructional template in language, serving as a generalizable form. Similar to the formulation in (Kim et al., 2024), each task either allows linguistic or object-level variations while maintaining consistent behavioral semantics, or accommodates behavioral variations under the same environmental conditions. For example, the $i$-th stage task $\mathcal{T}_i$ might include instructions such as "*clean an apple and put in the cabinet*", "*clean a towel and put in the washing machine*", and other variations that share similar behavioral patterns, which the agent is expected to follow. The observation captures the currently perceptible information from the environment, rendering the partially-observable nature of embodied task planning. The observation is represented as a set of triplets, each consisting of a source entity, a relation, and a target entity, e.g., *(apple, on, table)*. The rationale set $\mathcal{Z} = \{z_k\}_{k=1}^{N}$ contains key elements essential for embodied task planning (Yao et al., 2023; Choi et al., 2024), such as target object locations and sub-goals.

Formally, let $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_H\}$ be the sequence of adaptation tasks and $\mathbf{SR}(\pi, \mathcal{T}_i) \in [0, 1]$ the task success rate of policy $\pi$ on task $\mathcal{T}_i$. The objective of continual adaptation for embodied planning is

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \sum_{i=1}^{H} \mathbf{SR}(\pi, \mathcal{T}_i), \tag{1}$$

i.e., the policy that maximizes the cumulative task success rate across all $H$ stages.

### 3.2 Continual task adaptation with pre-trained models

To continuously adapt to tasks while leveraging the knowledge embedded in pre-trained models, recent works adopt parameter-efficient tuning modules in LMs (Liu et al., 2024; Schmied et al., 2023). Following this approach, we implement an embodied agent policy using Low-Rank Adaptation (LoRA) (Hu et al., 2022), where trainable adapter parameters $\theta$ are integrated with frozen LM parameters $\theta_{\text{LM}}$, i.e., $\pi(\cdot; \theta_{\text{LM}}, \theta)$. For simplicity, we use the notation $\pi(\cdot; \theta)$. Furthermore, we structure the embodied agent using separate modules for reasoning and planning, corresponding to a reasoning-policy and a planning-policy, each formed by distinct adapters $\theta_z$ and $\theta_p$, respectively. The reasoning-policy is responsible for rationale generation, while the planning-policy produces actions based on the generated rationales.

## 4 ASC Framework

To address the challenge of embodied task adaptation across sequential stages, we present the ASC framework comprising of (i) *amortizing self-correction* and (ii) *rationale-wise test-time scaling*, as depicted in Figure 1, where each stage receives few-shot demonstrations $\mathcal{D}$. (i) For adaptation, we first select a base reasoning-policy by evaluating the similarity between rationales generated by previously learned policies and those in the current stage's demonstrations. The selected one is used to initialize the reasoning-policy for the current adaptation stage. Then, the reasoning-policy is fine-tuned through jointly optimizing CoT reasoning and reflexive reasoning objectives. CoT reasoning is trained using rationales derived from demonstrations, while reflexive reasoning is formulated as the correction of base rationales generated by the selected base reasoning-policy, guided by feedback. The planning-policy predicts actions, conditioned on incrementally accumulated rationales. (ii) For inference, we dynamically scale the CoT reasoning process based on the planning-policy's confidence. The following two subsections describe how these two procedures are performed within a single stage with demonstrations $\mathcal{D}_i$ to enable task adaptation, where the stage notation $i$ is omitted for brevity.

### 4.1 Amortizing Self-Correction

**Base reasoning-policy selection.** At the beginning of each adaptation stage, we select a reasoning-policy $\pi_z$ learned from the previous stage to serve as the base reasoning-policy. Such base reasoning-policy is chosen by evaluating the similarity between rationales $z_k \in \mathcal{Z}$ in the demonstrations and the rationales $\bar{z}_k$ generated by each candidate reasoning-policy formed by an adapter $\bar{\theta}_z \in \Theta$ in

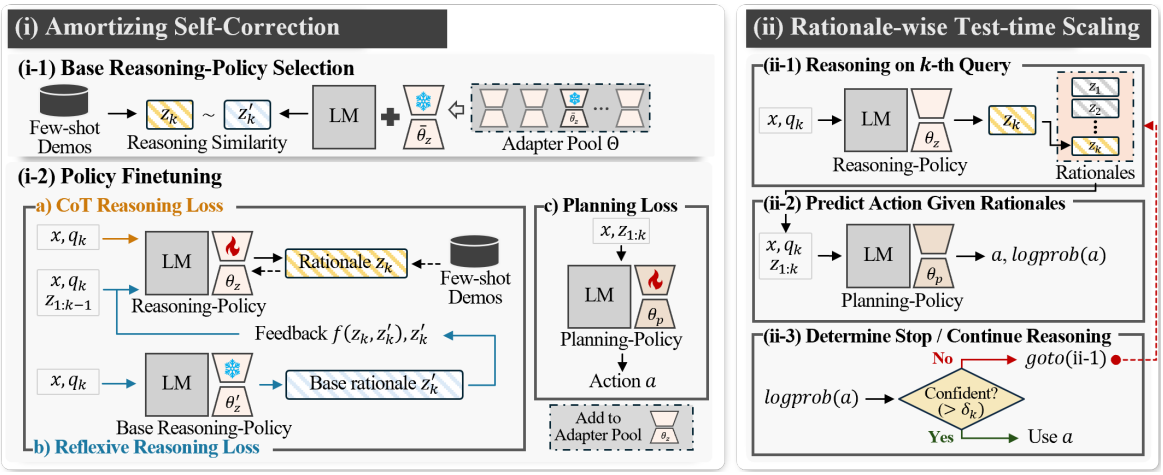



Figure 1: ASC framework: In (i-1), the most relevant reasoning-policy is selected from the adapter pool to serve as the base one. In (i-2), the reasoning-policy is then finetuned through joint optimization on CoT reasoning and reflexive reasoning losses. (a) The former loss is supervised by multi-step rationales in few-shot demonstrations, and (b) the latter loss is formalized as the correction of base rationales generated by the base reasoning-policy. In (ii), the CoT reasoning depth is dynamically adjusted based on the planning-policy's confidence in predicted actions.

the adapter pool $\Theta$. Note that the adapter pool contains the adapters learned up to the current stage. Then, the reasoning-policy maximizing the sentence embedding similarity SIM is chosen as

$$\theta'_z = \underset{\bar{\theta}_z \in \Theta}{\operatorname{argmax}} \sum_{(x,\mathcal{Z}) \in \mathcal{D}} \sum_{k=1}^{N} \mathrm{SIM}\left(z_k, \bar{z}_k\right), \quad (2)$$

where $\bar{z}_k \sim \pi_z(\cdot|x, q_k; \bar{\theta}_z)$. Here and in what follows, $x = (\mathcal{T}, o, h)$ denotes a triple of task $\mathcal{T}$, observation $o$, and action history $h$, and $q_k$ is the query for $k$-th rationale generation. The base reasoning-policy $\pi_z(\cdot; \theta'_z)$ is used to generate base rationales for subsequent correction and to initialize the adapter for the current stage.

**CoT reasoning loss.** To equip the reasoning-policy with CoT reasoning capabilities, we train it to generate rationales segment by segment. Then, the loss for CoT reasoning $\mathcal{L}_{\mathrm{CoT}}(\theta_z)$ is defined as

$$\mathop{\mathbb{E}}_{(x,\mathcal{Z}) \sim \mathcal{D}} \left[ \sum_{k=1}^{N} -\log \pi_z(z_k|x, q_k; \theta_z) \right]. \quad (3)$$

**Reflexive reasoning loss.** In addition to CoT reasoning, the reasoning-policy is trained to incrementally correct base rationales $z'_k \sim \pi_z(\cdot|x, q_k; \theta'_z)$ using feedback, focusing on one specific rationale at each correction step. The feedback $f(z_k, z'_k)$ is provided in natural language, categorized as "Major revision", "Moderate revision", and "Minor revision". This categorization is determined based on the sentence similarity $\mathrm{SIM}(z_k, z'_k)$ between the base rationale $z'_k$ and its corresponding rationale $z_k$ from the demonstrations. Then, the losss for reflexive reasoning $\mathcal{L}_{\mathrm{reflexive}}(\theta_z)$ is defined as

$$\mathop{\mathbb{E}}_{(x,\mathcal{Z}) \sim \mathcal{D}} \left[ \sum_{k=1}^{N} -\log \pi_z(z_k|z'_k, x, c_k; \theta_z) \right], \quad (4)$$

where $c_k = (q_k, f(z_k, z'_k), z_{1:k-1})$, and $z_{1:k-1} = \{z_1 \ldots, z_{k-1}\}$ denotes the sequence of previous rationales, and $z_{1:0} = \emptyset$.

This amortizing self-correction extends beyond mere CoT learning by systematically incorporating feedback-driven adjustment into reflexive reasoning steps, which identify discrepancies in prior knowledge. This enables the agent to further enhance task-specific knowledge, effectively improving its CoT reasoning capabilities. The final loss for the reasoning-policy $\pi_z$ is then defined as

$$\mathcal{L}_{\mathrm{reasoning}}(\theta_z) = \mathcal{L}_{\mathrm{CoT}}(\theta_z) + \mathcal{L}_{\mathrm{reflexive}}(\theta_z). \quad (5)$$

**Planning loss.** To predict actions based on the rationales, we optimize the planning-policy $\pi_p$ via an action reconstruction loss, where the policy is incrementally conditioned on the first $k$ rationales. This encourages the planning-policy to integrate partial reasoning signals. Then, the loss for planning-policy $\mathcal{L}_{\mathrm{planning}}(\theta_p)$ is defined as

$$\mathop{\mathbb{E}}_{(x,a,\mathcal{Z}) \sim \mathcal{D}} \left[ \sum_{k=1}^{N} -\log \pi_p(a|x, z_{1:k}; \theta_p) \right], \quad (6)$$

where $x = (\mathcal{T}, o, h)$. At the end of the adaptation, the adapter $\theta_z$ is added to the adapter pool $\Theta$ for

**Algorithm 1** Amortizing Self-Correction

**Input:** few-shot demos. $\{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_H\}$
Initialize adapter pool $\Theta = \emptyset$
**for** each adaptation stage $i \leftarrow 1, ..., H$ **do**
  Select base reasoning-policy $\theta'_z$ using (2)
  Initialize adapters $\theta_p, \theta_z \leftarrow \theta'_z$,
  **while** not converged **do**
    Sample a batch of $\{(\mathcal{T}, o, a, \mathcal{Z})\} \sim \mathcal{D}_i$
    Sample $z'_k \sim \pi_z(\cdot|x, q_k; \theta'_z)$
    Update reasoning-policy $\pi_z(\cdot; \theta_z)$ by (5)
    Update planning-policy $\pi_p(\cdot; \theta_p)$ by (6)
  Add adapters to adapter pool $\Theta \leftarrow \Theta \cup \{\theta_z\}$

use in subsequent adaptation stages. Algorithm 1 lists the adaptation procedures of ASC.

### 4.2 Rationale-wise test-time scaling

As the full set of rationales may be unnecessary for action prediction, depending on task complexity (Yao et al., 2023), we devise a test-time scaling mechanism in which the CoT reasoning depth is dynamically adjusted. At each CoT reasoning step $k$, the planning policy's confidence in its predicted action, conditioned on the first $k$ rationales $z_{1:k}$, is evaluated to determine whether to terminate or proceed to the next CoT reasoning step. Specifically, if the confidence on the predicted action does not exceed a predefined threshold $\delta_k$ at $k$-th reasoning step, an additional rationale $z_{k+1}$ is generated; otherwise, the corresponding action $a$ is used. Thus, rationale-wise test-time scaling is performed as

$$\text{TTS}(k) = \begin{cases} \text{generate } z_{k+1} & \text{if } s_k \leq \delta_k \\ \text{use } a & \text{otherwise,} \end{cases} \quad (7)$$

where $s_k = \log \pi_p(a|x, z_{1:k}; \theta_p)/|a|$, and $|a|$ is the token length of action $a$. The threshold is computed for each $k$ using the demonstrations by measuring the mean log-probability of the ground-truth action when the trained planning-policy is conditioned on the rationales $z_{1:k}$. This allows for efficient use of computational resources by adaptively scaling the CoT reasoning depth. The procedure for rationale-wise test-time scaling is provided in Algorithm 3.

## 5 Experiments

### 5.1 Experiment settings

**Environments.** For evaluation, we use VirtualHome (Puig et al., 2018) and ALFWorld (Shridhar et al., 2021). To configure the continual task adaptation setups, we follow the approach outlined in Kim et al. (2024). Specifically, in **Behavior Incremental Learning (Beh-IL)**, the agent is tasked to incrementally learn new behaviors, while in **Environment Incremental Learning (Env-IL)**, the agent is tasked to incrementally learn to perform behaviors in novel indoor scenes. We evaluate performance on two task categories. In the **Seen** category, instructions and indoor scenes are identical to those in the demonstrations, with variations only in the initial positions of objects. In the **Unseen** category, both instructions and indoor scenes are varied. Further details are in Appendix A.

**Datasets.** For default few-shot setting, we use 20 expert demonstrations per adaptation stage, while more constrained scenarios use only 5 demonstrations. For rationale annotation, we use GPT-4o-mini (Achiam et al., 2023).

**Evaluation metrics.** **Success rate (SR)** measures the fraction of episodes the agent completes successfully. **Goal success rate (GC)** measures the fraction of satisfied sub-goals out of all given goals. For both, we report the average performance achieved by the agent after each adaptation stage.

**Baselines.** We compare with several LM-based planning and continual task adaptation baselines. **(i) ReAct** (Yao et al., 2023) interleaves rationale generation and action prediction for embodied planning. **(ii) SayCan** (Ahn et al., 2022) integrates action feasibility into ReAct. **(iii) TAIL-Distill** (Liu et al., 2024) employs task-specific adapters for each adaptation stage. We implement two variants for comparison: **TAIL-Action**, which is trained only on state-action pairs from demonstrations; **TAIL-Distill**, which additionally incorporates rationales as ASC. **(iv) SeqFT-Distill** (Shridhar et al., 2023) uses CoT reasoning to distill step-by-step rationales, with each distilled policy for the current stages initializing the next stage. **(v) CAMA-Distill** (Kim et al., 2024) replays action logits from previous stages and adjusts their update weights and fine-tunes with rationales. Note that ReAct and SayCan use the same rationale-annotated demonstrations as ASC, but as in-context examples, while the other baselines (except for TAIL-Action) exploit these demonstrations for CoT reasoning distillation. At inference, all methods rely solely on CoT reasoning without explicit reflection. Additionally, we compare against **(vi) LLM-Planner** (Huang et al., 2023), which uses LLMs prompted with expert demonstrations as in-context examples.

**Implementation details.** For finetuning LM-based policies, we employ Qwen2.5-0.5B (Yang

| Method | Behavior Incremental Learning (Beh-IL) | | | | Environment Incremental Learning (Env-IL) | | | |
|---|---|---|---|---|---|---|---|---|
| | Seen | | Unseen | | Seen | | Unseen | |
| | SR (%) | GC (%) | SR (%) | GC (%) | SR (%) | GC (%) | SR (%) | GC (%) |
| **Benchmark: VirtualHome** | | | | | | | | |
| LLM-Planner (GPT-4o) | 81.63±1.63 | 91.79±0.87 | 76.44±1.90 | 90.11±0.99 | 88.38±1.43 | 92.88±0.80 | 70.26±2.14 | 80.31±1.51 |
| ReAct | 12.06±1.40 | 30.10±1.58 | 8.65±0.00 | 26.31±1.38 | 20.07±1.89 | 40.46±1.67 | 13.96±1.67 | 33.23±1.63 |
| SayCan | 46.50±2.16 | 65.25±1.62 | 37.96±2.04 | 56.76±1.70 | 60.00±1.89 | 73.99±1.37 | 48.64±1.50 | 67.11±1.08 |
| TAIL-Action | 55.00±2.29 | 69.58±1.74 | 33.45±2.22 | 52.45±1.94 | 61.56±1.85 | 78.24±1.23 | 36.21±1.42 | 58.93±1.11 |
| TAIL-Distill | 60.25±2.20 | 73.44±1.66 | 46.29±2.25 | 62.68±1.90 | 71.88±1.70 | 83.98±1.11 | 43.78±1.45 | 63.91±1.10 |
| SeqFT-Distill | 65.75±2.20 | 79.09±1.52 | 48.68±2.30 | 64.11±1.92 | 78.13±1.53 | 87.66±1.00 | 57.89±1.43 | 73.04±1.43 |
| CAMA-Distill | 59.13±2.22 | 71.28±1.74 | 44.28±2.28 | 59.60±1.96 | 77.81±1.56 | 87.69±0.96 | 43.46±1.45 | 63.27±1.10 |
| **ASC w/o TTS** | 76.13±1.91 | 83.87±1.39 | 60.29±2.32 | 69.13±1.97 | 85.94±1.29 | 91.56±0.84 | 62.81±2.58 | 75.37±1.02 |
| **ASC** | **81.38**±1.74 | **85.81**±1.36 | **64.03**±2.29 | **71.60**±1.94 | **94.06**±0.76 | **96.13**±0.52 | **71.99**±1.27 | **80.56**±0.97 |
| **Benchmark: ALFWorld** | | | | | | | | |
| LLM-Planner (GPT-4o) | 91.75±0.98 | 94.63±0.71 | 86.75±1.83 | 88.88±1.50 | 92.58±1.07 | 93.95±0.81 | 95.63±0.92 | 95.78±0.68 |
| ReAct | 0.00±0.00 | 4.34±0.36 | 0.00±0.00 | 2.24±0.21 | 0.00±0.00 | 1.07±0.19 | 0.00±0.00 | 0.70±0.13 |
| SayCan | 22.81±1.15 | 47.89±1.41 | 0.63±0.21 | 23.28±1.53 | 21.88±1.40 | 44.08±1.67 | 4.38±0.00 | 22.50±1.59 |
| TAIL-Distill | 53.91±1.65 | 59.69±1.44 | 39.53±2.16 | 46.95±2.01 | 58.79±2.01 | 68.13±1.64 | 31.41±2.45 | 46.40±2.17 |
| TAIL-Action | 56.11±1.78 | 68.71±1.44 | 37.97±2.50 | 55.72±2.12 | 47.56±2.13 | 63.23±1.69 | 24.63±2.24 | 43.06±2.05 |
| SeqFT-Distill | 64.30±1.70 | 71.85±1.44 | 47.97±2.42 | 57.98±2.16 | 61.91±2.06 | 70.71±1.72 | 39.22±2.47 | 51.24±2.23 |
| CAMA-Distill | 53.20±1.64 | 59.05±1.48 | 38.59±2.27 | 45.95±2.11 | 55.76±2.08 | 65.37±1.70 | 34.38±2.46 | 46.05±2.24 |
| **ASC w/o TTS** | 81.64±1.46 | 85.21±1.24 | 64.84±2.52 | 72.67±2.34 | 73.24±1.80 | 78.87±1.45 | 51.09±2.58 | 63.19±2.21 |
| **ASC** | **85.70**±1.34 | **88.37**±1.15 | **70.78**±2.41 | **77.38**±1.99 | **82.52**±1.57 | **85.61**±1.28 | **56.41**±2.56 | **66.12**±2.14 |

Table 1: Continual 20-shot adaptation performance: The baselines and ASC are evaluated in two setups (Beh-IL and Env-IL). The continual adaptation is structured over four stages, where each has 20 demos. We report results over four random seeds, evaluating each task with 4 episodes in VirtualHome and 8 episodes in ALFWorld, where episodes differ in their initial configurations. The SR and GC are reported in 95% confidence intervals. The best performance in each case among small LM-based approaches is highlighted in **bold**. Except for LLM-Planner with GPT-4o, all methods use the same LM of Qwen2.5-0.5B.

| Method | VirtualHome Beh-IL | | | | ALFWorld Env-IL | | | |
|---|---|---|---|---|---|---|---|---|
| | Seen | | Unseen | | Seen | | Unseen | |
| | SR (%) | GC (%) | SR (%) | GC (%) | SR (%) | GC (%) | SR (%) | GC (%) |
| TAIL-Distill | 53.87±2.11 | 69.88±1.62 | 38.54±2.12 | 58.45±1.81 | 33.57±1.81 | 49.18±1.62 | 29.64±2.06 | 45.40±1.95 |
| SeqFT-Distill | 49.96±2.21 | 63.99±1.71 | 41.35±2.17 | 62.38±1.75 | 35.54±1.86 | 51.93±1.67 | 28.66±2.20 | 45.34±2.04 |
| CAMA-Distill | 45.65±2.19 | 65.27±2.12 | 36.42±2.12 | 57.86±1.79 | 29.55±1.57 | 44.17±1.46 | 21.61±1.68 | 38.30±1.72 |
| **ASC w/o TTS** | 66.33±2.14 | 77.77±1.57 | 52.27±2.25 | 66.33±1.92 | 57.86±2.01 | 68.42±1.67 | 48.21±2.58 | 60.10±2.16 |
| **ASC** | **73.68**±1.92 | **80.99**±1.47 | **54.56**±2.29 | **67.38**±1.97 | **68.48**±1.89 | **76.09**±1.52 | **53.66**±2.55 | **64.20**±2.12 |

Table 2: Continual 5-shot adaptation performance

et al., 2024) with LoRA adapters (Hu et al., 2022). We also use a language embedding model of paraphrase-MiniLM-L6-v2 (Reimers and Gurevych, 2019) and TF-IDF score (Ramos et al., 2003) to compute the sentence similarity for base reasoning-policy selection and feedback provision.

### 5.2 Main results

**Continual 20-shot adaptation.** Table 1 shows the continual few-shot adaptation performance of ASC and the baselines under two different setups (Beh-IL and Env-IL) in VirtualHome and ALFWorld. Here, the continual adaptation stage is set to 4. ASC w/o TTS refers to ASC without rationale-wise test-time scaling in Section 4.2. As shown, ASC consistently yields robust performance across all cases. In the seen category, it achieves average improvements of 18.39% in SR and 11.65% in GC over the most competitive baseline SeqFT-Distill, and in the unseen category, ASC outperforms SeqFT-Distill by 17.36% in SR and 12.32% in GC on average. When compared to LLM-Planner utilizing GPT-4o, ASC exhibits a modest performance gap, 9.57% in SR and 9.59% in GC, despite only using a 0.5B-parameter model. In contrast, SeqFT-Distill suffers substantially larger drops of 27.45% in SR and 21.58% in GC.

In addition, we validate the effectiveness of rationale-wise test-time scaling, as ASC achieves slightly higher performance over its variant (ASC w/o TTS), with an average SR increase of 6.68% in the seen category and 6.05% in the unseen. We conjecture that due to limited data at each stage, certain

| Method | ALFWorld Beh-IL | | ALFWorld Env-IL | | Generated Rationales |
|---|---|---|---|---|---|
| | Seen | Unseen | Seen | Unseen | |
| **ASC** | **85.70**±1.34 | **70.78**±2.41 | **82.52**±1.57 | **56.41**±2.56 | 49% |
| w/o TTS | 81.64±1.46 | 64.84±2.52 | 73.24±1.80 | 51.09±2.58 | 100% |
| w/o TTS & *w/ self-correct* | 83.83±1.36 | 66.09±2.51 | 75.10±1.76 | 53.59±2.59 | 200% |
| Self-Correction | 56.48±1.68 | 44.06±2.31 | 63.18±2.03 | 37.66±2.53 | 200% |

Table 3: Computational efficiency in ASC measured by SR (%)

rationales might be under-optimized. Thus, allowing the model to terminate reasoning early when confidence is sufficiently high can make agent's decisions more reliably.

In these experiments, ReAct exhibits the lowest performance due to the limited reasoning capabilities of small LMs. SayCan achieves better performance by integrating action feasibility into embodied planning. In contrast, TAIL-Distill and SeqFT-Distill leverage LLM-generated rationales to distill CoT reasoning capabilities, resulting in improved performance. While CAMA-Distill preserves knowledge acquired from previous stages via logit replay, it still underperforms compared to ASC. This is because ASC effectively transfers reasoning knowledge via amortizing self-correction, which strengthens CoT reasoning by correcting the prior knowledge in response to novel tasks.

**Continual 5-shot adaptation.** In Table 2, we evaluate the performance under a more constrained few-shot setting, using 5 demonstrations per stage. Here, the continual adaptation stage is set to 7. As shown, ASC outperforms SeqFT-Distill by 28.33% higher SR and 16.75% higher GC in the seen category and 19.11% higher SR and 11.93% higher GC in the unseen. This result highlights the superiority of ASC under severely limited supervision.

### 5.3 Analysis and ablation studies

**Does ASC enable computationally efficient yet effective inference?** In Table 3, we evaluate *w/ self-correct*, a variant of ASC that performs explicit reflection on the initially generated rationales at inference. We also compare against Self-Correction (Welleck et al., 2023), which is trained to correct its *own* generated rationales (unlike ASC, which is trained to correct rationales produced by *prior* policies). As shown, explicit reflection yields modest gains over ASC w/o TTS, with a 2.19% SR increase in the seen category and 1.86% in the unseen, but at the cost of generating twice as many rationales. ASC achieves superior performance across all cases, outperforming Self-Correction by 25.78% SR in the seen category and 22.74% in the unseen. The limited effectiveness of explicit correction is attributed to the lack of precise feedback available at inference time, and the results highlight that amortizing self-correction offers greater benefits than correcting a model's own rationales. Moreover, ASC delivers clear computational efficiency, reducing rationale generation to one-quarter of that required by Self-Correction. This efficiency gain arises because ASC eliminates the need for explicit reflection and dynamically adjusts CoT depth.

**Does reflexive reasoning contribute to forward transfer?** In Table 4, we evaluate two ablated variants of ASC in ALFWorld. *w/o reflexive* uses the base reasoning-policy as the initialization for the current stage's policy, but is optimized without the reflexive reasoning loss in (4). *w/o base* does not leverage the base reasoning-policy in any form. ASC, with the reflexive reasoning loss, achieves relative gains in SR of 15.78% and 16.87% for the seen and unseen categories of ALFWorld Beh-IL, and 17.36% and 11.25% on those of ALFWorld Env-IL, respectively, over the *w/o reflexive* variant. Furthermore, ablating the base reasoning-policy initialization (i.e., the *w/o base* variant) incurs, on average, an additional performance degradation in SR of 6.73% for the seen and 8.83% for the unseen category relative to *w/o reflexive*. This reveals that ASC effectively leverages knowledge acquired from prior tasks through reflexive reasoning and base reasoning-policy selection.

| Method | ALFWorld Beh-IL | | ALFWorld Env-IL | |
|---|---|---|---|---|
| | Seen | Unseen | Seen | Unseen |
| **ASC** | **85.70**±1.34 | **70.78**±2.41 | **82.52**±1.57 | **56.41**±2.56 |
| *w/o reflexive* | 69.92±1.64 | 53.91±2.64 | 65.16±1.96 | 45.16±2.61 |
| *w/o base* | 65.16±1.72 | 47.66±2.58 | 56.47±2.02 | 33.75±2.46 |

Table 4: Forward knowledge transfer in ASC

**How does rationale-wise test-time scaling adapt to the complexity of an instruction?** In Table 5, we report the average plan length required for task completion and the percentage of the rationale $z_k$ at

| Instruction Template | Plan Length | $z_1$ | $z_2$ | $z_3$ | $z_4$ | $z_5$ | Tokens Reduced |
|---|---|---|---|---|---|---|---|
| TURNON | 3.80 | 49.4% | 4.5% | 14.8% | 7.2% | 24.1% | 51.1% |
| OPEN | 3.78 | 48.8% | 7.0% | 5.3% | 12.4% | 26.5% | 48.7% |
| PLACEON | 6.63 | 35.7% | 11.6% | 9.8% | 18.6% | 24.3% | 44.4% |
| PUTIN | 6.95 | 38.1% | 7.8% | 3.5% | 12.7% | 37.8% | 40.0% |

Table 5: Rationale-wise test-time scaling with respect to instruction complexity

which reasoning is terminated. We also report the percentage of reduction in reasoning tokens. For this, we analyze instruction templates in VirtualHome. As shown, for tasks requiring shorter plans (e.g., TURNON and OPEN), the reasoning process often stops at earlier rationales. In contrast, more complex tasks (e.g., PUTIN and PLACEON) tend to require late-stage rationales such as $z_4$ and $z_5$, which focus on sub-goal decomposition and next-step justification. Moreover, the reasoning tokens are reduced to 46.05% on average. This suggests that the test-time reasoning control dynamically adjusts the depth, enabling more efficient inference.

Additionally, Figure 14 (Appendix C) shows the effect of rationale-wise test-time scaling by applying it to SeqFT-Distill. As shown, performance rather decreases for the baseline, as it is trained to predict actions based on the entire rationale set. In contrast, ASC employs the loss in (6), which is optimized to predict actions conditioned on incrementally generated rationales, enabling effective decision-making even from partial rationales.

**Does ASC enable data-efficient adaptation?** In Table 6, we compare against SeqFT-Distill and TAIL-Distill, which is most competitive comparisons, under two data augmentation settings in ALFWorld. $2\times$ denotes training with twice the number of expert demonstrations, each augmented with a single set of rationales. *aug* denotes using the original number of expert demonstrations, but augmenting each demonstration with two sets of rationales. Note that ASC is trained with the original number of expert demonstrations with a single set of rationales. As shown, despite using less reasoning data, ASC outperforms the augmentation-based variants: it improves SR by 14.84% and 15.47% on the seen and unseen categories of ALFWorld-Beh-IL, and by 17.87% and 15.47% on those of ALFWorld-Env-IL, respectively, over the SeqFT-Distill (*aug*) setting. Moreover, relative to the SeqFT-Distill ($2\times$), ASC achieves a slight average SR improvement of 8.79% for the seen and 7.50% for the unseen category, despite using only half the data. This strong data efficiency of ASC is attributed to the amortizing self-correction strategy which effectively transfers prior knowledge.

| Method | ALFWorld Beh-IL | | ALFWorld Env-IL | |
|---|---|---|---|---|
| | Seen | Unseen | Seen | Unseen |
| **ASC** | **85.70**±1.34 | **70.78**±2.41 | **82.52**±1.57 | **56.41**±2.56 |
| SeqFT-Distill (2×) | 78.67±1.43 | 63.28±2.49 | 71.97±1.79 | 48.91±2.62 |
| TAIL-Distill (2×) | 66.56±1.56 | 50.16±2.37 | 61.72±1.95 | 39.53±2.49 |
| SeqFT-Distill (*aug*) | 70.86±1.63 | 55.31±2.54 | 64.65±1.96 | 40.94±2.60 |
| TAIL-Distill (*aug*) | 62.19±1.73 | 48.28±2.37 | 61.04±1.98 | 40.63±2.48 |

Table 6: Data-efficient adaptation in ASC

**Does selecting the most similar prior reasoning-policy yield better performance?** Table 7 shows two variants of ASC that use alternative base reasoning-policy selection strategies under the 5-shot setting. *Argmin* selects the policy with the lowest rationale similarity, while *Random* simply uses the random one. These variants do not employ test-time scaling, thereby isolating the effect of base reasoning-policy selection on the quality of generated rationales. As shown, ASC w/o TTS which selects the most similar one yields the highest performance, achieving an average SR increase of 5.67% in the seen category and 5.90% in the unseen compared to *Argmin*. Since the current stage's reasoning policy is initialized from the chosen base reasoning-policy, selecting the most similar predecessor provides a strong prior.

| Method | VirtualHome Beh-IL | | ALFWorld Env-IL | |
|---|---|---|---|---|
| | Seen | Unseen | Seen | Unseen |
| *Argmin* | 60.23±2.09 | 49.12±2.17 | 51.07±1.95 | 43.30±2.47 |
| *Random* | 62.35±2.04 | 50.79±2.19 | 54.64±1.94 | 42.64±2.45 |
| **ASC w/o TTS** | **66.33**±2.14 | **52.27**±2.25 | **57.86**±2.01 | **48.21**±2.58 |

Table 7: Effect of base policy selection strategy of ASC

**Can ASC be extended to support backward transfer?** ASC can be seamlessly extended to support backward transfer through memory-based rehearsal (Rolnick et al., 2018; Wan et al., 2024). To demonstrate this, we introduce ASC+Reh, a variant that retains demonstrations across adaptation stages. After the final stage, ASC+Reh refines earlier policies using the combined CoT and reflexive reasoning loss in (5). For each stage's policy, we se-

lect the most semantically relevant policy from the learned pool (excluding itself) as the base policy, and train it on the original demonstrations for that stage together with task-relevant demonstrations accumulated in the rehearsal buffer. Table 8 reports results under the continual 5-shot adaptation setup in VirtualHome Beh-IL. Incorporating the rehearsal mechanism improves performance, yielding SR gains of 5.06% in the seen category and 10.68% in the unseen category. These results demonstrate that ASC can effectively leverage memory-based rehearsal to enhance backward transfer.

| Method | Seen | | Unseen | |
|---|---|---|---|---|
| | SR (%) | GC (%) | SR (%) | GC (%) |
| **ASC** | 70.16±2.04 | 78.83±1.58 | 47.99±2.20 | 63.91±1.96 |
| **ASC+Reh** | **75.22**±1.82 | **81.33**±1.48 | **58.67**±2.31 | **68.79**±1.98 |

Table 8: ASC with rehearsal mechanism

## 6 Conclusion

In this paper, we presented the ASC framework tailored for efficient task adaptation of LM-based embodied agents under limited demonstrations. Specifically, our amortizing self-correction strategy enables effective forward knowledge transfer across adaptation stages, by jointly optimizing CoT reasoning and reflexive reasoning objectives. Notably, it facilitates robust CoT reasoning at inference without requiring explicit reflection steps. Furthermore, the rationale-wise test-time scaling mechanism focuses on sufficiently confident rationales, improving both efficient planning but also improved overall performance.

## Limitations

ASCoperates in an append-only manner, maintaining a separate adapter for each adaptation stage. As a result, the size of the policy pool grows linearly with the number of stages, which may become burdensome over extended task streams. This overhead is substantially mitigated by our use of LoRA adapters, where each stage introduces only 0.2% additional parameters relative to the 0.5B backbone LM. Nevertheless, scaling to substantially longer streams may require explicit mechanisms for consolidating the pool, such as merging semantically similar adapters or pruning redundant ones. We leave such extensions to future work.

## Ethical Considerations

LM-powered embodied agents may have broader societal impacts, including unintended behaviors or the amplification of existing biases. Additionally, as our framework builds on publicly available pretrained LMs, it may introduce incorrect or fabricated information in generated rationales. To mitigate such risks, we maintain full transparency of our rationale annotation pipeline and experimental setup in Appendices A and B, enabling independent verification and replication.

## Acknowledgments

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT), (RS-2022-II220043, Adaptive Personality for Intelligent Agents, RS-2022-II221045, Self-directed multi-modal Intelligence for solving unknown, open domain problems, RS-2025-02218768, Accelerated Insight Reasoning via Continual Learning, RS-2025-25442569, AI Star Fellowship Support Program (Sungkyunkwan Univ.), RS-2026-25543726, Development of Leading Talent in Medical Domain-Specific Generative AI, RS-2026-25528384, Resource-Intensive AI Technologies Based on Sustainable GPU Integrated Platforms, RS-2019-II190421, Artificial Intelligence Graduate School Program (Sungkyunkwan University)), the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2026-25474409), IITP-ITRC (Information Technology Research Center) grant funded by the Korea government (MSIT) (IITP-2025-RS-2024-00437633, 10%), and by Samsung Electronics Co., Ltd.

## References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Micheal Ahn, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander Toshev, Vincent Vanhoucke, and 26 others. 2022. Do as I can, not as I say: Grounding language in robotic affordances. In *Proceedings of the 6th Conference on Robot Learning*.

Wonje Choi, Woo Kyung Kim, Minjong Yoo, and Honguk Woo. 2024. Embodied cot distillation from LLM to off-the-shelf agents. In *Proceedings of the 41st International Conference on Machine Learning*.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 81 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, and 3 others. 2023. Palm-e: An embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*.

Chongkai Gao, Haichuan Gao, Shangqi Guo, Tianren Zhang, and Feng Chen. 2021. CRIL: continual robot imitation learning via generative and prediction model. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Rishi Hazra, Pedro Zuidberg Dos Martires, and Luc De Raedt. 2024. Saycanpay: Heuristic planning with large language models using learnable domain knowledge. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *Proceedings of the 10th International Conference on Learning Representations*.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022a. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *Proceedings of the 39th International Conference on Machine Learning*.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2023. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. 2022b. Inner monologue: Embodied reasoning through planning with language models. In *Proceedings of the 6th Conference on Robot Learning*.

Byeonghwi Kim, Minhyuk Seo, and Jonghyun Choi. 2024. Online continual learning for interactive instruction following agents. In *Proceedings of the 12th International Conference on Learning Representations*.

Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Li E Li, Ruohan Zhang, and 1 others. 2025. Embodied agent interface: Benchmarking llms for embodied decision making. In *Proceedings of the 39th Annual Conference on Neural Information Processing Systems*.

Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Li Erran Li, Ruohan Zhang, Weiyu Liu, Percy Liang, Li Fei-Fei, Jiayuan Mao, and Jiajun Wu. 2024. Embodied agent interface: Benchmarking llms for embodied decision making. In *Proceedings of the 38th Annual Conference on Neural Information Processing Systems*.

Ryan Liu, Jiayi Geng, Addison J Wu, Ilia Sucholutsky, Tania Lombrozo, and Thomas L Griffiths. 2025. Mind your step (by step): Chain-of-thought can reduce performance on tasks where thinking makes humans worse. In *Proceedings of the 42nd International Conference on Machine Learning*.

Zuxin Liu, Jesse Zhang, Kavosh Asadi, Yao Liu, Ding Zhao, Shoham Sabach, and Rasool Fakoor. 2024. TAIL: task-specific adapters for imitation learning with large pretrained models. In *Proceedings of the 12th International Conference on Learning Representations*.

Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine. 2023. Adaplanner: Adaptive planning from feedback with language models. In *Proceedings of the 37th Annual Conference on Neural Information Processing Systems*.

Sam Powers, Eliot Xing, Eric Kolve, Roozbeh Mottaghi, and Abhinav Gupta. 2022. CORA: benchmarks, baselines, and metrics as a platform for continual reinforcement learning agents. In *Proceedings of the 1st Conference on Lifelong Learning Agents*.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. 2018. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Juan Ramos and 1 others. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. New Jersey, USA.

Nils Reimers and Iryna Gurevych. 2019. 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*.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. 2018. Experience replay for continual learning. In *Proceedings of the 32rd Annual Conference on Neural Information Processing Systems*.

Thomas Schmied, Markus Hofmarcher, Fabian Paischer, Razvan Pascanu, and Sepp Hochreiter. 2023. Learning to modulate pre-trained models in RL. In *Proceedings of the 37th Annual Conference on Neural Information Processing Systems*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Proceedings of the 37th Annual Conference on Neural Information Processing Systems*.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2023. Distilling reasoning capabilities into smaller language models. In *Findings of the Association for Computational Linguistics*.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. 2021. Alfworld: Aligning text and embodied environments for interactive learning. In *Proceedings of the 9th International Conference on Learning Representations*.

Weikang Wan, Yifeng Zhu, Rutav Shah, and Yuke Zhu. 2024. LOTUS: continual imitation learning for robot manipulation through unsupervised skill discovery. In *Proceedings of the 12th International Conference on Robotics and Automation*.

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. 2023. Generating sequences by learning to self-correct. In *Proceedings of the 11th International Conference on Learning Representations*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, and 1 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *Proceedings of the 11th International Conference on Learning Representations*.

Nick Yeung and Christopher Summerfield. 2012. Metacognition in human decision-making: confidence and error monitoring. *Philosophical Transactions of the Royal Society B: Biological Sciences*.

Minjong Yoo, Jinwoo Jang, Wei-Jin Park, and Honguk Woo. 2024. Exploratory retrieval-augmented planning for continual embodied instruction following. In *Proceedings of the 38th Annual Conference on Neural Information Processing Systems*.

# A Benchmark environments

## A.1 VirtualHome

VirtualHome (Puig et al., 2018) is a Unity-based simulation environment where an agent interacts with household objects to accomplish given instructions. There are 50 different house settings in VirtualHome, each with different room layouts and object positions. The environment features a wide variety of real-life objects, making it challenging for embodied agents.

**Instruction.** We configure four instruction templates, each with a unique goal: TURNON, OPEN, PLACEON, PUTIN. In Table 9, we provide an example for each instruction template.

| Instruction template | Example |
|---|---|
| TURNON | Turn on tv |
| OPEN | Open cabinet |
| PLACEON | Place apple on sofa |
| PUTIN | Put mug in microwave |

Table 9: Instruction templates in VirtualHome

**Observation.** We use a Sentence-BERT (Reimers and Gurevych, 2019) to retrieve the top-$k$ most relevant triples from the environment knowledge graph based on the given instruction. This setup follows similar practices in recent works in embodied agents (Choi et al., 2024; Yoo et al., 2024). In Figure 2, we provide an example of observation.

(character, inside, kitchen), (character, hold, none), (kitchen, adjacent, bedroom), (kitchen, adjacent, livingroom), (bathroom, adjacent, bedroom), (tv, inside, bedroom), (hanger, inside, bedroom), (powersocket, inside, kitchen), (wineglass, inside, kitchen), (kitchencabinet, inside, kitchen), (salmon, inside, kitchen) ...

Figure 2: Observation example for VirtualHome

**Action.** The available actions include: *walk*, *open*, *switch*, *grab*, *place on*, and *put in*. The first four actions take a single argument (e.g. *walk kitchen*, *grab apple*), while the last two actions require two arguments (e.g. *place apple on sofa*, *put mug in microwave*). In Table 10, we present the format of each action along with a corresponding example.

**Continual task adaptation setup.** We configure two continual adaptation setups similar to CAMA (Kim et al., 2024). For Behavior Incremental Learning (Beh-IL), the agent incrementally learns new behaviors; for Environment Incremental Learning (Env-IL), the agent incrementally learns to perform behaviors in novel indoor scenes.

| Format | Example |
|---|---|
| *walk* [object or room] | *walk* kitchen |
| *open* [object] | *open* cabinet |
| *switch* [object] | *switch* tv |
| *grab* [object] | *grab* apple |
| *place* [object] *on* [object] | *place* apple *on* sofa |
| *put* [object] *in* [object] | *put* mug *in* microwave |

Table 10: Format of each action in VirtualHome

For **Beh-IL**, we utilize the four instruction templates described above, assigning each as the primary behavior for a different learning stage. Experiments are conducted across four randomly ordered sequences of tasks, and evaluated under **Seen** and **Unseen** categories. For the Seen category, we use the same tasks and room layouts as those used for training; while for the Unseen category, we introduce variations in room layouts, instructions, and initial object positions. The specific sequences used for Beh-IL in VirtualHome are detailed in Figure 3. For the continual 5-shot adaptation setup, which spans seven stages, templates may recur across stages; however, the specific tasks instantiated from each template are disjoint across stages, ensuring that no task is revisited during adaptation.

1. TURNON → OPEN → PLACEON → PUTIN
2. OPEN → TURNON → PUTIN → PLACEON
3. PLACEON → OPEN → TURNON → PUTIN
4. PUTIN → OPEN → PLACEON → TURNON

Figure 3: Beh-IL adaptation sequences (VirtualHome)

For **Env-IL**, we utilize the diverse room layouts provided by VirtualHome, each featuring different object and furniture arrangements. Experiments are conducted across four randomly ordered sequences of these room layouts, and the Seen and Unseen categories are constructed in the same manner as in Beh-IL. The room sequences used for Env-IL in VirtualHome are detailed in Figure 4.

**Expert demonstrations.** To construct few-shot demonstrations, we emulate a rule-based expert that generates an optimal plan based on the given instruction. For each adaptation stage, we collect 20 expert demonstrations, varying the initial positions

1. ROOM20 → ROOM01 → ROOM34 → ROOM26
2. ROOM01 → ROOM20 → ROOM26 → ROOM34
3. ROOM34 → ROOM01 → ROOM20 → ROOM26
4. ROOM26 → ROOM01 → ROOM34 → ROOM20

Figure 4: Env-IL adaptation sequences (VirtualHome)

of the agent and objects, as well as the instructions. For the more constrained continual few-shot adaptation settings, we use 5 expert demonstrations.

### A.2 ALFWorld

ALFWorld (Shridhar et al., 2021) is a text-based environment designed for embodied task planning, featuring numerous household tasks brought from ALFRED (Shridhar et al., 2020) benchmark. It supports complex task compositions and is widely used to evaluate instruction-following agents under partially observable settings.

**Instruction.** We configure four instruction templates, each with a unique goal: HEAT, CLEAN, PICK&PLACE, PICK2&PLACE. In Table 11, we provide an example for each instruction template.

| Instruction template | Example |
|---|---|
| HEAT | Heat some plate in microwave and put it in/on shelf |
| CLEAN | Clean some fork in sinkbasin and put it in/on sidetable |
| PICK&PLACE | Put a keychain in/on armchair |
| PICK2&PLACE | Find two newspaper and put them in/on armchair |

Table 11: Instruction templates in ALFWorld

**Observation.** The observation settings are identical to VirtualHome in Section A.1. In Figure 5, we provide an example of observation.

(character, inside, bedroom), (character, hold, keychain 1) (character, close, armchair 1), (laptop 1, inside, armchair 1), (lightswitch 1, is, visible), (cellphone 3, inside, bed 1), (cd 2, inside, garbagecan 1), (pillow 1, inside, bed 1), (cd 1, inside, dresser 1), (alarmclock 2, inside, dresser 1), (cabinet 1, is, visible) ...

Figure 5: Observation example for ALFWorld

**Action.** The available actions include: *go to*, *take from*, *put in/on*, *heat with*, and *clean with*. The *go to* action takes a single argument specifying the destination (e.g. *go to shelf 1*), while the other actions require two arguments (e.g. *take potato 1 from countertop 3*, *clean spoon 1 with sinkbasin 1*). In Table 12, we provide the format of each action along with a corresponding example.

| Format | Example |
|---|---|
| *go to* [object] | *go to* shelf 1 |
| *take* [object] *from* [object] | *take* potato 1 *from* countertop 3 |
| *put* [object] *in/on* [object] | *put* newspaper 1 *in/on* ottoman 1 |
| *heat* [object] *with* [object] | *heat* potato 3 *with* microwave 1 |
| *clean* [object] *with* [object] | *clean* spoon 1 *with* sinkbasin 1 |

Table 12: Format of each action in ALFWorld

**Continual task adaptation setup.** Similar to VirtualHome, we configure two continual task adaptation setups in ALFWorld.

For **Beh-IL**, we utilize the four instruction templates, assigning each as the primary behavior for an adaptation stage. The specific sequences used for Beh-IL in ALFWorld are detailed in Figure 6.

1. HEAT → PICK2&PLACE → PICK&PLACE → CLEAN
2. PICK2&PLACE → CLEAN → HEAT → PICK&PLACE
3. PICK&PLACE → PICK2&PLACE → CLEAN → HEAT
4. CLEAN → HEAT → PICK2&PLACE → PICK&PLACE

Figure 6: Beh-IL adaptation sequence (ALFWorld)

For **Env-IL**, we utilize different room types in ALFWorld, including bathroom, bedroom, livingroom, and kitchen. The specific sequences used for Env-IL in ALFWorld are detailed in Figure 7. Similarly, for the continual 5-shot adaptation setup, which spans seven stages, room layouts may recur across stages; however, the tasks instantiated in each layout remain disjoint, ensuring that no task is revisited during adaptation.

1. BEDROOM → BATHROOM → LIVINGROOM → KITCHEN
2. BATHROOM → BEDROOM → KITCHEN → LIVINGROOM
3. LIVINGROOM → BATHROOM → BEDROOM → KITCHEN
4. KITCHEN → BATHROOM → LIVINGROOM → BEDROOM

Figure 7: Env-IL adaptation sequence (ALFWorld)

**Expert demonstrations.** To construct the training demonstrations, we utilize the expert provided in

ALFWorld. For each adaptation stage, we utilize 20 expert demonstrations, varying the initial positions of the agent and objects, as well as the instructions. For the more constrained continual few-shot adaptation setting, we use 5 expert demonstrations.

## B Implementation details

In this section, we provide the rationale annotation strategy and implementation details of ASC and the baselines. All experiments are conducted on a system equipped with an Intel(R) Core(TM) i9-10980XE CPU and an NVIDIA RTX A6000 GPU. We use GPT-4o-mini (Achiam et al., 2023) (MiT license) for rationale annotation and Qwen2.5-0.5B (Yang et al., 2024) (Apache 2.0 license) as the pre-trained LM of the policies. Both models are used solely for research purposes, consistent with their intended use.

### B.1 Rationale annotation

Rationales $\mathcal{Z} = \{z_1, ..., z_N\}$ are annotated through LLMs (e.g., GPT-4o-mini (Achiam et al., 2023)) for each observation and action pair $(o, a)$. We define a set of Markov Decision Process-featured queries $\mathcal{Q} = \{q_1, ..., q_N\}$, each designed to extract key elements required for embodied task planning, such as the location of the target object, the status of the agent, sub-goals, available actions, and expected returns (Choi et al., 2024). We set $N = 5$, generating one rationale per query; Table 13 summarizes the element extracted by each query.

| $\mathcal{Q}$ | **Key elements** |
|---|---|
| $q_1$ | physical location and status of the agent |
| $q_2$ | physical location and status of observations relevant to the task |
| $q_3$ | summarization of previous action history, if available |
| $q_4$ | sub-goals needed to complete the task |
| $q_5$ | reasoning about the next action to take |

Table 13: Key elements extracted by each query

We prompt the LLM $\Psi_{\text{LLM}}$ with queries $\mathcal{Q}$, along with the task $\mathcal{T}$, observation $o$, action history $h(= a_{1:t})$, and the current action $a(= a_t)$ to sample a rationale set, i.e., $\mathcal{Z} \sim \Psi_{\text{LLM}}(\mathcal{Q}, x, a)$, where $x = (\mathcal{T}, o, h)$. Next, we feed the policy $\pi(\cdot|\theta_{\text{LM}})$ (i.e., the pre-trained LM without any attached adapters) with the observation and rationale set to obtain logits over the available actions $\bar{a} \in \mathcal{A}$. If $a$ falls within the top-$k$ by logit value, we store the rationale set $\mathcal{Z}$ in the buffer $\mathcal{B}$. We repeat this procedure $I$ times with varying temperature and nucleus-sampling parameters to collect diverse pool of rationale sets.

$$\mathcal{B} = \bigcup_{i=1}^{I} \{\mathcal{Z} | a \in \text{top-}k_{\bar{a}\in\mathcal{A}}\,(\pi(\bar{a}|o, \mathcal{Z}; \theta_{\text{LM}}))\}, \quad \text{where } \mathcal{Z} \sim \Psi_{\text{LLM}}(\mathcal{Q}, x, a). \tag{8}$$

Subsequently, we query the policy with each rationale set in the buffer to obtain the logits of the ground-truth future plan $p(= a_{t:T})$, and select the set that yields the highest logit as the final rationale set for reasoning-policy training, i.e.

$$\mathcal{Z} = \underset{\mathcal{Z}^i \in \mathcal{B}}{\text{argmax}}\, \pi(p|\mathcal{T}, o, \mathcal{Z}^i; \theta_{\text{LM}}). \tag{9}$$

The entire procedure is repeated for each transition in $\mathcal{D}$, producing one annotated rationale set per sample. Algorithm 2 details the full procedure, and Figure 8 shows an example of rationales annotated by the LLM for VirtualHome benchmark.

**Algorithm 2** Rationale annotation

**Input:** demonstrations $\mathcal{D}$, LLM $\Psi_{\text{LLM}}$, pre-trained LM parameters $\theta_{\text{LM}}$
**for** each $(\mathcal{T}, o, a) \in \mathcal{D}$ **do**
  Initialize buffer $\mathcal{B} = \emptyset$
  **for** $i \leftarrow 1, ..., I$ **do**
    $\mathcal{Z} \sim \Psi_{\text{LLM}}(\mathcal{Q}, x, a)$
    **if** $a \in$ top-$k_{\bar{a}\in\mathcal{A}}(\pi(\bar{a}|o, \mathcal{Z}; \theta_{\text{LM}}))$ **then**
      $\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathcal{Z}\}$
  $\mathcal{Z} = \text{argmax}_{\mathcal{Z}^i \in \mathcal{B}}\, \pi(p|\mathcal{T}, o, \mathcal{Z}^i; \theta_{\text{LM}})$
  Augment $\mathcal{D}$ with $(\mathcal{T}, o, a, \mathcal{Z})$

### B.2 Baseline Methods

The baselines in our experiments fall into three categories: **[a]** LM-based embodied planning method (e.g., LLM-Planner, ReAct, SayCan), **[b]** continual embodied adaptation method (e.g., TAIL, SeqFT, CAMA), **[c]** self-correction method (e.g., Self-Correction). Below, we detail the implementation of each baseline.

**[a] LLM-Planner** (Huang et al., 2023) leverages the in-context learning capability of language models by prompting few-shot demonstrations. Specifically, two demonstrations are sampled based on the task similarity, which is computed by Jaccard distance. For inference, LLM-Planner employs

| | |
|---|---|
| **[Instruction]** | Place book on coffeetable |
| **[Observation]** | (character, inside, kitchen), (character, hold, book), (kitchen, adjacent, bedroom), (kitchen, adjacent, livingroom), (coffeetable, inside, livingroom), (book, is, closed), (character, close, paper), (pie, inside, kitchen), (facecream, inside, bathroom), (closet-drawer, inside, livingroom) ... |
| **[History]** | (step 1, walk bedroom), (step 2, walk kitchen), (step 3, walk book), (step 4, grab book) |
| **[Action]** | walk livingroom |
| **[Rationales]** | The character is currently inside the kitchen, holding a closed book. The coffeetable is located inside the livingroom, which is adjacent to the kitchen. Previously, the character walked from the bedroom to the kitchen, grabbed the book, and is now preparing to move it. To complete the instruction of placing the book on the coffeetable, the character needs to walk to the livingroom next. Since the livingroom is adjacent to the kitchen, the character can proceed there to fulfill the instruction. |

Figure 8: Example of rationales for VirtualHome

an action-level decoding strategy to select actions from the set of valid ones (Hazra et al., 2024). Figure 9 presents the prompt used for LLM-Planner.

**[a] ReAct** (Yao et al., 2023) generates rationales and actions in an interleaved manner. To allow the LM to better leverage its reasoning capabilities, we prompt it with few-shot in-context CoT demonstrations. We use the same prompt as in Figure 9, but with rationales additionally included for each in-context demonstration.

**[a] SayCan** (Ahn et al., 2022) incorporates an additional module that accounts for affordance scores when selecting actions. This module re-weights the action logits based on predicted affordance scores, ensuring the feasibility of plans and preventing the agent from selecting invalid actions in the current state. For simplicity, we replace this module by heuristically providing only executable actions.

**[b] TAIL** (Liu et al., 2024) employs parameter-efficient fine-tuning methods to address the continual task adaptation problem. For each adaptation stage, a LoRA adapter is newly initialized and trained on the demonstrations through supervised finetuning. Based on this, we implement two variants for comparison. **TAIL-Action** is trained solely on state-action pairs without utilizing rationale supervision. **TAIL-Distill** leverages rationales to supervise both reasoning and planning policies. The same CoT reasoning loss defined in Equation (3) for ASC is used to train the reasoning-policy for TAIL-Distill. At inference, rationales are generated segment-by-segment through the reasoning-policy, and subsequently used to generate actions through the planning-policy. The hyperparameter settings for TAIL are summarized in Table 14.

| Hyperparameter | Value |
|---|---|
| Total epochs | 80 |
| Batch size | 4 |
| Learning rate | 1.41e-5 |
| LoRA alpha | 32 |
| LoRA rank | 16 |

Table 14: Hyperparameter settings for TAIL

**[b] SeqFT-Distill** SeqFT-Distill uses the policy learned from the most recent adaptation stage as the initialization point, unlike TAIL, which randomly initializes model parameters for each adaptation stage, This ensures implicit knowledge transfer across stages through model parameters. For reasoning distillation, we employ both reasoning and planning policies as ASC. We use the same hyperparameter settings as in Table 14 for SeqFT-Distill.

**[b] CAMA-Distill** (Kim et al., 2024) addresses the continual learning problem in embodied tasks by introducing a method to update stored past logits in episodic memory. To accommodate reasoning distillation, we adopt the same two-tier policy architecture as ASC. A small subset of previous demonstrations, along with their corresponding logits, is retained across adaptation stages and used to supervise the policy alongside the current stage's demonstrations. We use the same hyperparameter settings as in Table 14 for CAMA-Distill.

**[c]Self-Correction** (Welleck et al., 2023) learns to iteratively refine imperfect generations. To adapt this approach in our setting, we employ LMs with

**Prompt used for LLM-Planner in ALFWorld**

**Interact with a household to solve a task. Following are the only actions available:**

go to [recep]: move to a receptacle/location

take [obj] from [recep]: pick up an object from a receptacle

put [obj] in/on [recep]: place an object inside or on top of a receptacle

heat [obj] with [recep]: heat an object using a receptacle

clean [obj] with [recep]: clean an object using a receptacle

**Here are some example demonstrations:**

*[Few-shot Demonstration 1]*

Your task is to: put a pencil in/on desk.

Observation: (character, inside, bedroom), (book 3, inside, desk 1), (laptop 1, inside, desk 1), (pencil 2, inside, garbagecan 1), (pencil 1, inside, sidetable 1), (alarmclock 1, inside, desk 1), (remotecontrol 1, inside, desk 1), (alarmclock 2, inside, desk 1), (book 2, inside, desk 1)

$>$ Action: go to sidetable 1

*[Few-shot Demonstration 2]*

⋮

*[Few-shot Demonstration N]*

**Here is the task:**

Your task is to: put a pencil in/on desk.

Observation: (character, inside, bedroom), (pen 1, inside, desk 1), (laptop 1, inside, desk 1), (pencil 2, inside, sidetable 1), (pencil 3, inside, shelf 1), (keychain 3, inside, desk 1), (box 1, inside, desk 1), (pen 2, inside, desk 1), (book 1, inside, desk 1), (cd 1, inside, garbagecan 1), (cd 2, is, visible), (blinds 1, is, visible), (chair 1, is, visible)

$>$ Action:

Figure 9: Example prompt used for LLM-Planner in ALFWorld

adapters for both the initial rationale generator and the corrector. Feedback for correction is obtained in the same way as our setup. The generated rationale is compared with the ground-truth rationales in the demonstrations using a combined similarity score based on language embeddings and TF-IDF scores. At inference, the model first produces an initial rationale and then performs explicit self-correction. The surrogate feedback is provided by retrieving the rationale in the demonstrations that is most semantically similar to the current state.

### B.3 ASC (ours)

The ASC framework consists of two main processes: (i) amortizing self-correction, and (ii) rationale-wise test-time scaling. (i) For adaptation, we first select the most relevant previously learned reasoning-policy to serve as the base one. Then, an LM-based policy is jointly trained via CoT and reflexive reasoning objectives from few-shot demonstrations, where CoT reasoning is supervised by rationale distillation and reflexive reasoning by base rationale correction. We use the same hyperparameter settings as in Table 14 for ASC. (ii) For inference, the policy solely relies on CoT reasoning, with its depth dynamically adjusted according to the model's confidence in predicted actions.

The threshold $\delta_k$ for rationale-wise test-time scaling in (7) is derived from the mean and standard deviation of the log-probability of the ground-truth action when the planning-policy is conditioned on partial rationales $z_{1:k}$ as

$$\begin{aligned}\delta_k \;&=\; \mathbb{E}_{(x,a,\mathcal{Z})\sim\mathcal{D}}\Big[\log \pi_p(a|x, z_{1:k}; \theta_p)/|a|\Big] \\ &+\; \lambda \cdot \mathrm{std}_{(x,a,\mathcal{Z})\sim\mathcal{D}}\Big[\log \pi_p(a|x, z_{1:k}; \theta_p)/|a|\Big]\end{aligned} \tag{10}$$

where $\lambda$ is a hyperparameter corresponding to a desired percentile cutoff, converted into z-score. Thresholds are computed separately at each adaptation stage, yielding a distinct $\delta_k$ for every reasoning step. In our experiment, we set $-0.524$ for VirtualHome (i.e., allowing the top 70% of scores to pass the threshold) and $0.0$ for ALFWorld (i.e., allowing 50% of scores to pass the threshold). This choice reflects the fact that ALFWorld tasks typically involve a larger number of observations with redundant and repeated entities compared to VirtualHome. Therefore, ALFWorld requires a stricter

**Algorithm 3** ASC framework

**Input:** environment Env, task $\mathcal{T}$
$o_0 \leftarrow$ Env.reset()
$done \leftarrow false,\ t \leftarrow 0,\ h \leftarrow \emptyset$
**while** not done **do**
  $x_t \leftarrow (\mathcal{T}, o_t, h)$
  **for** $k \leftarrow 1, ..., N$ **do**
    $z_{t,k} \sim \pi_z(\cdot|x_t, q_k; \theta_z)$
    $a_t \sim \pi_p(\cdot|x_t, z_{t,1:k}; \theta_p)$
    **if** $\log \pi_p(a_t|x_t, z_{t,1:k}; \theta_p)/|a_t| \geq \delta_k$ **then**
      break
  $o_{t+1} \leftarrow$ Env.step$(a_t)$
  $h \leftarrow h \cup \{a_t\},\ t \leftarrow t+1$

threshold than VirtualHome. A sensitivity analysis of this threshold is provided in Section C.8.

The procedure of rationale-wise test-time scaling is listed in Algorithm 3. Prompts used for CoT reasoning, reflexive reasoning, and the planning policy are shown in Figures 10, 11, and 12, respectively.

**Prompt used for CoT reasoning (query 1) in ASC for ALFWorld**

Instruction: heat some cup in microwave and put it in/on sidetable

Observation: (character, inside, kitchen), (microwave 1, is, visible), (cup 2, inside, countertop 1), (cup 1, inside, sinkbasin 1), (peppershaker 1, inside, sidetable 1), (pot 2, inside, stoveburner 4), (bread 1, inside, countertop 1), (soapbottle 1, inside, garbagecan 1), (spoon 1, is, visible)

Previous Actions: No action history.

**[Query 1] You should provide exactly 1 sentence response that incorporate: *physical location and status of the character***

> Response:

Figure 10: Prompt used for CoT reasoning in ASC for ALFWorld

**Prompt used for reflexive reasoning in ASC for ALFWorld**

Instruction: heat some cup in microwave and put it in/on sidetable

Observation: (character, inside, kitchen), (microwave 1, is, visible), (cup 2, inside, countertop 1), (cup 1, inside, sinkbasin 1), (peppershaker 1, inside, sidetable 1), (pot 2, inside, stoveburner 4), (bread 1, inside, countertop 1), (soapbottle 1, inside, garbagecan 1), (spoon 1, is, visible)

Previous Actions: No action history.

Reasoning Trace: The character is currently inside the kitchen. The cup 2 is located on countertop 1, while the microwave 1 is also visible in the kitchen. There is no previous action history to summarize. To complete the instruction, the character needs to first go to countertop 1, retrieve cup 2, heat it in the microwave, and then place it on sidetable 1.

Rationale: Heat some cup in microwave and put it in/on sidetable.

**There are many errors in the Think. You need a major revision in the Think. You should provide exactly 1 sentence response that only incorporate: *reasoning for what should do next.***

> Response:

Figure 11: Prompt used for reflexive reasoning in ASC for ALFWorld

**Prompt used for planning-policy used in ASC for VritualHome**

**Following are the only actions available:**

walk [object or room]: walk to object or room

open [object]: open an object

switch [object]: switch on an object

grab [object]: grab an object

place [object] on [recep]: put holding on top of an object

put [object] in [recep]: put holding inside of an object

**In order to complete the given instruction, what should be the next immediate action?**

Instruction: Open stove

Observation: (character, inside, kitchen), (character, hold, none), (kitchen, adjacent, bedroom), (kitchen, adjacent, livingroom), (bathroom, adjacent, bedroom), (stove, is, closed), (stove, inside, kitchen), (stove, is, off), (toaster, is, off), (closetdrawer, inside, bedroom)

Previous Actions: No action history.

Rationale: The character is currently inside the kitchen, situated near the closed stove. The stove, which needs to be opened, is located inside the kitchen and adjacent to a bedroom. There is no previous action history to consider. To complete the instruction, the character must first walk to the stove. Thus, the logical next action should be walk to stove.

> Action:

Figure 12: Prompt used for planning-policy in ASC for VirtualHome

# C Additional Experiments

## C.1 Effect of segment-wise reasoning

In Figure 13, we compare two variants of ASC with different levels of granularity in their reasoning processes. *Full* learns to generate and correct the entire rationale in a single inference step, while *Chunk-wise* processes multiple rationales (two or three) at a time. In contrast, ASC adopts a segment-wise approach, processing one rationale at an inference step, thus enabling the most fine-grained control over the reasoning process. As shown, finer-grained reasoning consistently leads to performance improvements. This is particularly beneficial for smaller LMs, whose limited capacity benefits from step-by-step guidance.

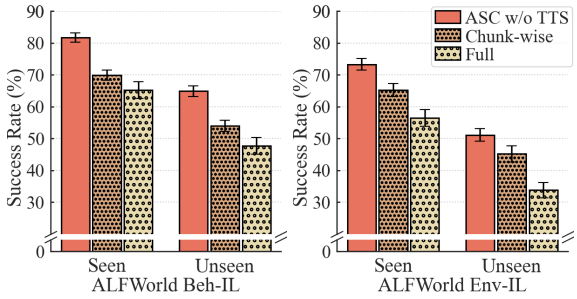



Figure 13: Effect of segment-wise reasoning

## C.2 Effect of rationale-wise test-time scaling

Figure 14 analyzes the effect of rationale-wise test-time scaling mechanism in ASC. To isolate its contribution, we apply the same mechanism to the SeqFT-Distill baseline. We observe that, while TTS improves the performance of ASC, it instead degrades that of the SeqFT-Distill baseline. As discussed in Section 5.2, this is because SeqFT-Distill is trained to predict the entire rationale as a single inference, leaving on natural granularity at which rationale-wise scaling can operate. Whereas, ASC is trained in a segment-wise manner, allowing TTS to contribute meaningfully to performance.

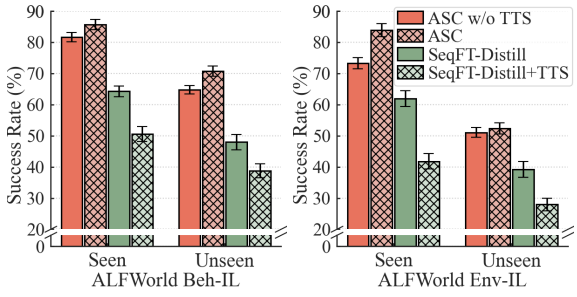



Figure 14: Effect of rationale-wise test-time scaling

## C.3 Planning Efficiency

To assess the planning efficiency beyond the SR reported in Table 1, we report Normalized Plan Efficiency (NPE) defined as optimal plan length divided by executed plan length (higher is better, and 1.0 indicates an optimal plan) While SR reflects task completion, NPE captures how efficiently the agent completes the given task. A higher NPE indicates more efficient planning, with a value 1.0 representing a perfectly optimal plan.

In Table 15, we report NPE of ASC and the baselines under continual task adaptation setups in VirtualHome and ALFWorld. As shown, ASC consistently achieves higher NPE compared to the baselines, demonstrating its superiority in generating not only successful but also efficient plans. This is attributed to the enhanced robustness of CoT reasoning, enabled by reflexive reasoning that corrects prior knowledge to internalize more precise task knowledge across learning stages.

| Method | Beh-IL Seen | Beh-IL Unseen | Env-IL Seen | Env-IL Unseen |
|---|---|---|---|---|
| **Benchmark: VirtualHome** | | | | |
| TAIL-Action | 0.79±0.02 | 0.75±0.02 | 0.80±0.01 | 0.77±0.01 |
| TAIL-Distill | 0.80±0.01 | 0.76±0.02 | 0.81±0.01 | 0.73±0.01 |
| SeqFT-Distill | 0.78±0.01 | 0.72±0.02 | 0.84±0.01 | 0.79±0.01 |
| CAMA-Distill | 0.79±0.01 | 0.78±0.01 | 0.80±0.01 | 0.75±0.01 |
| **ASC w/o TTS** | 0.84±0.01 | **0.84±0.01** | 0.84±0.01 | **0.81±0.01** |
| **ASC** | **0.85±0.01** | **0.84±0.01** | **0.87±0.01** | **0.81±0.01** |
| **Benchmark: ALFWorld** | | | | |
| TAIL-Action | 0.88±0.01 | 0.84±0.01 | 0.87±0.01 | 0.88±0.01 |
| TAIL-Distill | 0.89±0.01 | 0.81±0.01 | 0.91±0.01 | 0.88±0.01 |
| SeqFT-Distill | 0.91±0.01 | 0.84±0.01 | 0.91±0.01 | 0.89±0.01 |
| CAMA-Distill | 0.89±0.01 | 0.88±0.01 | 0.89±0.01 | 0.86±0.01 |
| **ASC w/o TTS** | **0.95±0.00** | 0.93±0.01 | 0.94±0.01 | 0.90±0.01 |
| **ASC** | **0.95±0.00** | **0.94±0.01** | **0.95±0.00** | **0.92±0.01** |

Table 15: Planning efficiency measured by NPE

## C.4 Continual Task Adaptability

To further assess continual task adaptability, we evaluate ASC and the baselines (TAIL-Distill and SeqFT-Distill) on the complete set of tasks from all stages at each individual adaptation stage. Here, we compare with ASC w/o TTS to isolate the effect of test-time scaling, thereby highlighting the effectiveness of amortizing self-correction alone. For tasks from stages not yet encountered, evaluation is performed using the most recently learned policy. As shown, the success rate (SR) increases roughly linearly for all baselines and ASC as the number of stages grows, indicating that the agents incrementally acquire new behaviors and adapt to

| Method | Stage 1 | | Stage 2 | | Stage 3 | | Stage 4 | |
|---|---|---|---|---|---|---|---|---|
| | Seen | Unseen | Seen | Unseen | Seen | Unseen | Seen | Unseen |
| TAIL-Distill | 20.83±0.92 | 15.21±0.92 | 33.55±0.87 | 26.27±1.08 | 49.80±1.74 | 36.38±1.83 | 60.25±2.20 | 46.29±2.25 |
| SeqFT-Distill | 19.13±0.91 | 13.89±0.87 | 32.75±1.00 | 23.81±1.13 | 49.33±1.67 | 37.26±1.83 | 62.17±2.22 | 46.31±2.22 |
| **ASC w/o TTS** | **24.63**±0.91 | **19.78**±0.98 | **38.25**±1.01 | **29.42**±1.30 | **56.63**±1.61 | **44.79**±1.96 | **76.13**±1.91 | **60.29**±2.32 |

Table 16: Continual Task Adaptability

novel scenes. However, the performance gain of ASC becomes more pronounced with increasing adaptation stages, achieving a 5.50% improvement at stage 2 and a 13.96% improvement at stage 4 on the seen category in VirtualHome Beh-IL compared to SeqFT-Distill. This advantage stems from our amortizing self-correction, which enables more effective forward transfer of previously acquired knowledge to new tasks.

## C.5 Scaling to Larger Models

To validate that ASC scales naturally to larger LMs, we evaluate ASC and SeqFT-Distill on VirtualHome Beh-IL using **Qwen2.5-1.5B** and **Qwen2.5-3B** in Table 17, Consistent with 0.5B results in Table 1, ASC outperforms the SeqFT-Distill across both model sizes. In the seen category, ASC achieves SR gains of 14.99% and 13.75% on the 1.5B and 3B models, respectively; in the unseen category, the gains reach 16.29% and 14.43%. These results demonstrate that ASC scales robustly, maintaining its advantage across models sizes.

| Method | Seen | | Unseen | |
|---|---|---|---|---|
| | SR (%) | GC (%) | SR (%) | GC (%) |
| **Model: Qwen2.5-0.5B** | | | | |
| SeqFT-Distill | 65.75±2.15 | 79.09±1.52 | 48.68±2.30 | 64.11±1.92 |
| **ASC** | **81.38**±1.74 | **85.81**±1.36 | **64.03**±2.28 | **71.60**±3.03 |
| **Model: Qwen2.5-1.5B** | | | | |
| SeqFT-Distill | 70.13±2.01 | 79.88±1.49 | 52.34±2.30 | 64.41±1.96 |
| **ASC** | **85.12**±1.52 | **89.28**±1.14 | **68.64**±2.12 | **75.16**±1.87 |
| **Model: Qwen2.5-3B** | | | | |
| SeqFT-Distill | 73.25±2.08 | 83.28±1.43 | 57.94±2.34 | 69.22±1.92 |
| **ASC** | **87.00**±1.37 | **91.91**±0.91 | **72.37**±2.09 | **78.07**±1.80 |

Table 17: Performance with varying model sizes

## C.6 Experiments with Compositional Tasks

To evaluate ASC on more challenging tasks, we construct compositional tasks in VirtualHome, where the agent must complete a sequence of two or three instructions, such as "Turn on computer, and turn on radio" (composition of two instructions) or "Put apple in fridge, and turn on stove, and place paper on bed" (composition of three instructions). As shown in Table 18, ASC outperforms SeqFT-Distill, achieving a 7.29% SR gain in the seen category and a 15.65% gain in the unseen category. These results highlight the capability of ASC to adapt to complex, compositional tasks.

| Method | Seen | | Unseen | |
|---|---|---|---|---|
| | SR (%) | GC (%) | SR (%) | GC (%) |
| SayCan | 22.08±0.98 | 48.18±0.90 | 15.84±0.75 | 45.18±0.80 |
| SeqFT-Distill | 48.12±1.64 | 69.85±1.49 | 31.00±1.15 | 57.94±1.29 |
| **ASC w/o TTS** | 50.59±1.67 | 73.68±1.63 | 40.56±1.02 | 64.12±1.31 |
| **ASC** | **55.41**±1.55 | **77.13**±1.56 | **46.65**±0.92 | **71.28**±1.09 |

Table 18: Performance on compositional tasks

## C.7 Analysis on Reasoning Depth

To verify that the benefit of rationale-wise TTS stems from its adaptivity rather than from simply reasoning longer, we compare it against fixed-depth variants, where the CoT reasoning depth is fixed at $k$ rationales for every task regardless of its complexity. As shown in Table 19, ASC with rationale-wise TTS outperforms all fixed-depth variants on VirtualHome Beh-IL, exceeding the strongest variant by 1.88% and 3.74% in the seen and unseen categories, respectively. Notably, performance across fixed depths is non-monotonic: no single $k$ performs best in both categories, and the worst setting ($k = 4$) falls 11.38% and 14.94% below ASC. These results indicate that the optimal reasoning depth varies across tasks, and that adaptively determining it from planning-policy confidence is more effective than committing to any fixed depth.

| Method | Seen SR (%) | Unseen SR (%) |
|---|---|---|
| ASC (TTS) | **81.38** | **64.03** |
| ASC ($k = 1$) | 77.62 | 57.74 |
| ASC ($k = 2$) | 79.50 | 56.06 |
| ASC ($k = 3$) | 77.00 | 54.89 |
| ASC ($k = 4$) | 70.00 | 49.09 |
| ASC ($k = 5$) | 76.13 | 60.29 |

Table 19: Analysis on Reasoning Depth

### C.8 Sensitivity to Rationale-wise Test-time Scaling Threshold

To assess the sensitivity to the threshold $\delta_k$ used for rationale-wise test-time scaling in (7), we vary the threshold levels in VirtualHome Beh-IL by adjusting $\lambda$ in (10). In Table 20, *Low* denotes the default threshold setting used in our main experiments. As shown, applying test-time scaling generally yields higher performance than not applying it. Setting the threshold too low (e.g., *Very Low*) prematurely halts CoT reasoning, resulting in a performance drop of 3.76% in the seen category and 5.84% in the unseen compared to the *low* (default) setting. Conversly, setting the threshold too high (e.g. *Very High*) also leads to degradation, with drops of 6.26% in the seen category and 2.49% in the unseen. We conjecture that limited data at each adaptation stage leaves some rationales under-optimized, and forcing the model to "over think" compounds errors and degrades overall performance (Liu et al., 2025). Nevertheless, ASC consistently outperforms all baselines in Table 1 across all threshold values, while being more computationally efficient. Notably, even under the worst-performing threshold condition (Very High), ASC still exceeds the most competitive baseline, SeqFT-Distill, by 9.37% in the seen category and 12.41% in the unseen category.

| Threshold Level | Seen | Unseen |
|---|---|---|
| w/o TTS | 76.13±1.91 | 60.29±2.32 |
| Very Low | 77.62±1.88 | 57.74±2.29 |
| Low (Default) | **81.38**±1.74 | **63.58**±2.28 |
| High | 79.25±1.82 | 61.35±2.01 |
| Very High | 75.12±1.98 | 61.09±2.29 |

Table 20: Sensitivity analysis on rationale-wise test-time scaling thresholds