Embodied CoT Distillation From LLM To Off-the-shelf Agents

Wonje Choi [1] **Woo Kyung Kim** [1] **Minjong Yoo** [1] **Honguk Woo** [1]

Abstract

We address the challenge of utilizing large language models (LLMs) for complex embodied tasks, in the environment where decision-making systems operate timely on capacity-limited, off-the-shelf devices. We present DEDER, a framework for decomposing and distilling the embodied reasoning capabilities from LLMs to efficient, small language model (sLM)-based policies. In DEDER, the decision-making process of LLM-based strategies is restructured into a hierarchy with a reasoning-policy and planning-policy. The reasoning-policy is distilled from the data that is generated through the embodied in-context learning and self-verification of an LLM, so it can produce effective rationales. The planning-policy, guided by the rationales, can render optimized plans efficiently. In turn, DEDER allows for adopting sLMs for both policies, deployed on off-the-shelf devices. Furthermore, to enhance the quality of intermediate rationales, specific to embodied tasks, we devise the embodied knowledge graph, and to generate multiple rationales timely through a single inference, we also use the contrastively prompted attention model. Our experiments with the ALFRED benchmark demonstrate that DEDER surpasses leading language planning and distillation approaches, indicating the applicability and efficiency of sLM-based embodied policies derived through DEDER.

1. Introduction

In embodied AI, significant advancements have been made in applying large language models (LLMs) to task planning. For example, SayCan (Brohan et al., 2023) combines LLMs' reasoning capabilities with a reinforcement learning (RL)-based affordance model to interpret task instructions and deduce executable robotic skills in the environment. Several works (Huang et al., 2022; Wu et al., 2023; Song et al., 2023; Singh et al., 2023) explore the grounding of LLMs to the environment through prompting based on sensory data, reference trajectories, and available skills. Recently, palm-e (Driess et al., 2023) expands the embodied reasoning abilities of LLMs to include multimodal data, such as visual observations of the environment. Yet, these approaches, which directly rely on LLMs for continual short-term decision-making, often encounter practical limitations in real-world applications, particularly when decision-making agents are required to operate on capacity-constrained, off-the-shelf devices. The high computational requirements of LLMs pose a significant challenge in such scenarios.

The direct end-to-end distillation of an LLM into a more compact, resource-efficient model, while it appears straightforward, might not be effective for complex embodied tasks (Dasgupta et al., 2023). This challenge stems from the requirements of a deep understanding on embodied task features, which inherently demand the long-horizon multi-step reasoning along with the ability to adapt to time-varying environment contexts. An embodied agent frequently encounters new and unseen environment information through its interaction with the surroundings. This continual exposure to a diverse range of environment conditions adds layers of complexity and variability, which in turn complicates the distillation process.

Our work is focused on the distillation of LLM-based policies for embodied tasks into off-the-shelf agents that are only capable of operating small language models (sLMs). We present DEDER, an innovative embodied distillation framework, designed to decompose and distill the embodied reasoning and decision-making procedures of LLM-based policies into two distinct small, more manageable models: reasoning-policy and planning-policy. The reasoning-policy focuses on understanding and interpreting task requirements and environment contexts, while the planning-policy concentrates on generating actionable plans based on the insights provided by the reasoning-policy. This division allows for the sophisticated functionalities of LLMs to be leveraged in a more resource-efficient manner, suitable for embodied agents with capacity-limited, off-the-shelf devices.

Achieving the reasoning-policy via LLM distillation

[1]Department of Computer Science and Engineering, Sungkyunkwan University, Suwon, Republic of Korea. Correspondence to: Honguk Woo <hwoo@skku.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

presents a unique challenge due to the hidden nature of reasoning processes within an LLM. We address this by employing the embodied Chain-of-Thought (CoT) and in-context learning, enhanced with self-verification, through the iterative use of the LLM. For the reasoning-policy, we employ the embodied knowledge graph (KG)-based prompting and the contrastively prompted attention model, integrated with an sLM. These two techniques improve the quality of rationale outcomes from the reasoning-policy, by integrating the environment information to a KG efficiently and representing the current context effectively. They also allow for a parallel structure for multiple rationale generation, thereby facilitating the timely task planning at runtime. The planning-policy exploits the distilled rationales to determine executable plans, addressing the practical need for actionable decision-making for complex tasks.

Using the ALFRED benchmark (Shridhar et al., 2020), our experiments exhibit the advantages of DEDER. The results demonstrate that the policy derived through DEDER significantly surpasses other baselines such as LLM-planner (Song et al., 2023) in zero-shot task planning scenarios. DEDER achieves a substantial improvement of 15.0% in seen task settings and 21.1% in unseen settings. Considering that DEDER employs an sLM at runtime instead of LLMs, the results clearly underline the exceptional adaptability of DEDER in handling new and unencountered environments.

Note that DEDER is the first framework to achieve sLM-based policies, which is resource-efficient yet comparable to LLM-based policies (i.e., the baselines in Section 5.1) in performance, for complex embodied tasks. The contributions of our work are summarized as follows.

- We present the novel framework DEDER, addressing the challenges of distilling LLMs' reasoning capabilities for embodied tasks to a small policy, readily deployed on capacity-limited, off-the-shelf devices.
- We devise the two-tier policy hierarchy in DEDER, through which the embodied reasoning process is decomposed and its knowledge can be distilled systematically to achieve a robust sLM-based policy.
- We develop the data construction process from LLMs for rationales specific to embodied tasks, exploring in-context learning and self-verification techniques.
- We implement the embodied KG and prompted attention model for sLM-based policies, to enhance the rationale quality across environment changes and facilitate rapid task planning.
- Through extensive experiments on ALFRED, we show DEDER's effectiveness and efficiency in achieving robust zero-shot performance for unseen embodied tasks.

2. Related Work

LLM-based Embodied Control. In the field of embodied control, there is a growing trend of utilizing LLMs for reasoning and execution of tasks in real-world settings (Brohan et al., 2023; Huang et al., 2022; Song et al., 2023). Our work aligns with this direction but sets itself apart by aiming to enable off-the-shelf devices to attain comparable embodied task performance without directly utilizing LLMs at runtime, instead focusing on tuning a smaller language model.

Embodied Policy Distillation. Recently, several works focused on distilling complex decision-making strategies, often derived from computationally intensive models, into compact and efficient ones suitable for resource-constrained environments. In (Sumers et al., 2023), knowledge is distilled from pre-trained vision-language models to supervise the language grounded skills of instruction-following agents. In (Jain et al., 2021), a two-stage training scheme is adopted for visual embodied agents. A relevant subset of policy distillation in RL is transferring the teacher policy in a supervised fashion (Yin & Pan, 2017). In particular, prior work concentrated on reducing the cross-entropy between the distributions of teacher and student policies (Parisotto et al., 2016; Schmitt et al., 2018). Our LLM-based policy distillation is also to minimize the divergence from the distribution of a teacher policy, which is an LLM, while exploring the unique two-tier hierarchy in decomposition and distillation of the LLM's reasoning capabilities.

Reasoning Capabilities of LLMs. Numerous studies investigated the reasoning capabilities of LLMs, exploring methods like retrieval-augmented in-context examples (Lewis et al., 2020; Ram et al., 2023), KG integration (Andrus et al., 2022; Baek et al., 2023), and CoT prompting (Wei et al., 2022; Wang et al., 2022). Recent research also demonstrated the effectiveness of distilling CoT processes from LLMs into sLMs (Wang et al., 2023; Li et al., 2023). Our work is in the same vein as LLM distillation, but specifically targets complex embodied tasks and uses decomposed distillation.

3. Problem Formulation

In RL, an environment for embodied agents is modeled as a Partially Observable Markov Decision Process (POMDP), represented by a tuple $(\mathcal{S}, \mathcal{A}, P, \mathcal{G}, \mathcal{H}, R, \Omega, \mathcal{O})$ (Song et al., 2023; Singh et al., 2023). Here, $s \in \mathcal{S}$ is a state space, $a \in \mathcal{A}$ is an action space, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a transition probability, $G \in \mathcal{G}$ is a goal space, $h \in \mathcal{H}$ is a high-level task description and $R : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \to \mathbb{R}$ is a reward function. The distinct aspect of embodied agents' environment lies in its nature of partial observations, featured as an observation space $o \in \Omega$ and a conditional observation probability $\mathcal{O} : \mathcal{S} \times \mathcal{A} \to \Omega$ (Sutton & Barto, 2018). This aspect accounts for the agents' limited perception, rendering

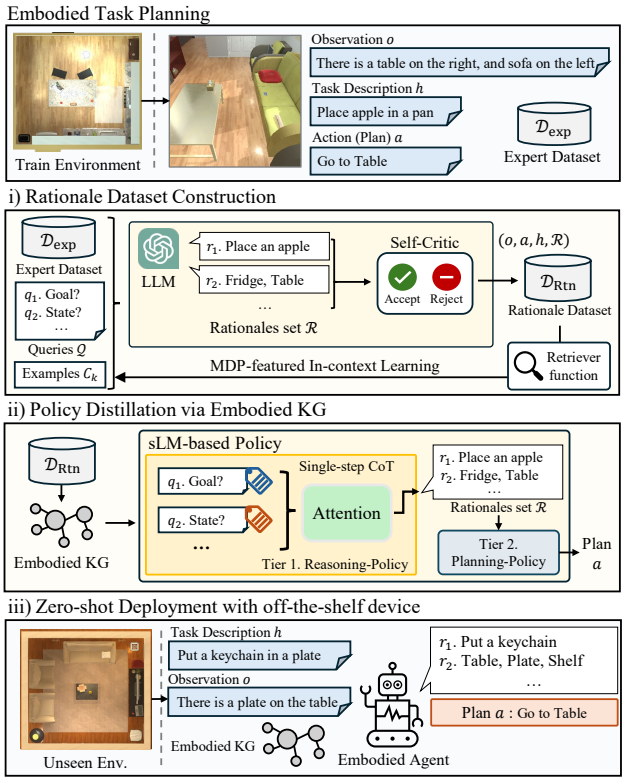


Figure 1. DEDER framework with three phases: (i) In rationale dataset construction phase, the MDP-featured in-context learning and self-critic function are employed to extract rationales from the LLM; (ii) In policy distillation phase, the sLM-based policy consisting of reasoning-policy and planning-policy is trained using the extracted rationale data; (iii) In zero-shot deployment, the distilled sLM-based policy is evaluated in unseen environments.

the decision-making complex and reflective of real-world situations. Our goal is to achieve a robust sLM-based policy Φ^*_{sLM} for capacity-limited, off-the-shelf devices, which is comparable to the capabilities in embodied task planning demonstrated by LLM-based policies Φ_{LLM}.

$$\Phi^*_{\text{sLM}} = \underset{\Phi_{\text{sLM}}}{\operatorname{argmax}}\, \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \Phi_{\text{sLM}}(o_t, h_t), G) - D(\Phi_{\text{LLM}}(o_t, h_t), \Phi_{\text{sLM}}(o_t, h_t))\right] \quad (1)$$

Note that D is a distance function such as Kullback-Leibler divergence (Kullback & Leibler, 1951) and γ is a discount factor of the environment.

4. Approach

For embodied tasks, it is essential for the agent to have reasoning capabilities to understand and interact with complex, dynamic environments. Yet, the simplification of the reasoning process is particularly necessary when employing an sLM-based policy, given the inherent limitations of sLMs due to their restricted model capacity. This can be achieved by integrating Markov Decision Process (MDP) features such as goal, state, observation, action, return-to-go, and sub-goal, which RL formulations specify, into the reasoning process (Chane-Sane et al., 2021; Hausknecht & Stone, 2015; Chen et al., 2021; Janner et al., 2021).

In this work, we refer to this type of environment information and MDP features as rationales, as they can function as justifications or hints that help to elaborate the reasoning behind plans. We leverage these rationales as a means to effectively distill the embodied reasoning capabilities from an LLM to small models, thereby achieving an sLM-based policy. For this distillation, we develop the DEDER framework comprising these phases: (i) rationale dataset construction, (ii) policy distillation via embodied KG, and (iii) zero-shot deployment and evaluation, as illustrated in Figure 1.

In the phase of rationale dataset construction, we harness the CoT scheme inherent in the usage of LLMs to extract rationales from expert transitions (i.e., series of action plans) in the environment. This is achieved through MDP-featured in-context learning, employing RL-specific queries as prompts that are defined by the properties of the MDP. In the subsequent phase of policy distillation, we establish an sLM-based policy structured in a two-tier hierarchy based on an embodied KG. It includes a reasoning-policy, which is trained to generate rationales in a single-step CoT optimized by behavior-based contrastive learning, as well as a planning-policy, which is learned to infer action plans through CoT prompting guided by these rationales. In the deployment phase, we evaluate distilled sLM-policy in a zero-shot manner for unseen environments in which task descriptions, object positions, and indoor scenes are changed.

4.1. Rationale Dataset Construction

Consider an expert dataset $\mathcal{D}_{\text{exp}} = \{\tau_i = (o_i, a_i, h_i)\}_i$, where each transition τ_i includes an observation o_i, action (plan) a_i, and high-level task description h_i for timesteps i. We expand the $\mathcal{D}_{\text{exp}}$ dataset to establish a rationale dataset $\mathcal{D}_{\text{Rtn}} = \{c_i = (o_i, a_i, h_i, \mathcal{R}_i)\}_i$, where each transition τ_i is supplemented with a rationale set $\mathcal{R} = \{r_j\}_{j=1}^m$. To obtain the rationale set specifically configured for given embodied tasks, we integrate MDP-featured in-context learning with the CoT prompting mechanism of an LLM. This involves iteratively prompting the LLM with a series of RL-specific queries, exploiting retrieval-augmented examples, similar to (Ram et al., 2023). Subsequently, the rationale set undergoes LLM's assessments, as discussed in (Sun et al., 2023), to be incorporated into the dataset $\mathcal{D}_{\text{Rtn}}$.

MDP-Featured In-Context Learning. To extract the rationales from the LLM using the transition τ, we continually update in-context examples in a retrieval-augmented

Task Description *h*: In the environment, various objects are positioned across different household areas. Assigned task for the embodied agent is to put apple in a pan …
Observation *o*: Currently visible objects include a dish-sponge, bread, apple, drawer, …
So far, the agent planned to 'go to the garbage-can', 'pick up the apple from the garbage-can'.
Action (Plan) *a*: Next plan is 'go to the stove-burner'.

Queries	Rationales
Query #1 (Goal): What is the task assigned?	**Rationale #1:** The assigned task involves placing an apple in pan and subsequently moving it to the dining-table.
Query #2 (State): Which objects are relevant to the task?	**Rationale #2:** Relevant objects for the task are an apple (in the fridge, on the garbage-can, and on the microwave), a pan (on the stove-burner), and a dining-table in the environment.
Query #3 (Observation): Among the currently visible objects, which one is relevant to the task?	**Rationale #3:** Currently, visible objects relative to the task include an apple and a garbage-can.
Query #4 (Action): What actions can the agent perform?	**Rationale #4:** Agent can interact with apple, garbage-can.
Query #5 (Action history): Summary of the agent's past planning actions?	**Rationale #5:** Agent went to the garbage-can and picked up the apple.
Query #6 (Return-to-go): Are there any steps left to be completed?	**Rationale #6:** Remaining tasks include moving to the pan and placing the pan with the apple on the dining-table.
Query #7 (Sub-goal): What is the most effective action to take now?	**Rationale #7:** The most efficient behavior now is to move to the stove-burner.

Figure 2. MDP-featured in-context learning in DEDER for rationale extraction from the LLM: the examples of inputs, queries (in red), and rationales (in blue) for the desired plan are presented, wherein MDP-aligned ones are specifically emphasized.

manner from dataset $\mathcal{D}_{\text{Rtn}}$. We use a retriever function $F : (\tau, \mathcal{C}) \mapsto \mathcal{C}_k$, as described in (Karpukhin et al., 2020). It takes a transition τ from $\mathcal{D}_{\text{exp}}$ and a set of tuples $\mathcal{C} = \{c_1, ..., c_n\}$ from $\mathcal{D}_{\text{Rtn}}$ as input, and retrieves the top-k most semantically relevant tuples from $\mathcal{C}$ for given τ, thus achieving an example set $\mathcal{C}_k$. The semantic relevance is calculated by the inner product between language embeddings of τ and c through the pre-trained contextual embedding model E. That is, we obtain relevance $S(\tau, c) = E(\tau)^\top E(c)$.

With the tuples $\mathcal{C}_k$, we then have the rationale set $\mathcal{R}$ sequentially by prompting the LLM Φ_{LLM} with a pre-defined set of RL-specific queries $\mathcal{Q} = \{q_1, ..., q_m\}$.

$$\mathcal{R} = \{r_l | r_l = \Phi_{\text{LLM}}(\mathcal{C}_k, \tau, \{r_j\}_{j<l},\ q_l)\} \tag{2}$$

Here, $\{r_j\}_{j<l}$ denotes a set of previously generated rationales for the questions preceding r_l. In this process, $\mathcal{C}_k$ is used to enhance the in-context learning of the LLM, as described in (Ram et al., 2023), enabling it to effectively respond to queries q_l. In specific, RL-specific queries are designed to extract MDP features, which are necessary for embodied task planning such as a goal, state, plan, observation, plan history, and sub-goal. The example of these queries and rationales is shown in Figure 2.

LLM as a Self-Critic Function. To ensure that the rationale set $\mathcal{R}$ aligns with the action plan a, we also use the LLM as a self-critic function. Specifically, we use a query q_{cri} to prompt the LLM to check whether the plan a can be induced solely from the extracted $\mathcal{R}$. In cases when $\mathcal{R}$ does not provide sufficient information for a, we start over by retrieving in-context examples. Otherwise, we incorporate the newly generated tuple $c = (o, a, h, \mathcal{R})$ to the dataset $\mathcal{D}_{\text{Rtn}}$. By employing this self-verification, we aim to gather rationales that encompass sufficient information to deduce plans in the expert transitions.

$$\mathcal{D}_{\text{Rtn}} = \{c_i | \Phi_{\text{LLM}}(q_{\text{cri}}, \mathcal{R}_i, a_i) = 1, c_i \in \mathcal{D}_{\text{Rtn}}\} \tag{3}$$

4.2. Policy Distillation via Embodied Knowledge Graph

To distill the reasoning capabilities of the LLM to an sLM-based policy Φ_{sLM} using the rationale dataset $\mathcal{D}_{\text{Rtn}}$, we structure the policy in a two-tier hierarchy. The first tier is a reasoning-policy Φ_{R}; it is responsible for inferring a rationale set from a given observation o, a task description h, and an embodied KG g. The second tier is a planning-policy Φ_{P}; it generates the plan, guided by the rationales from Φ_{R}.

$$\Phi_{\text{sLM}} = \Phi_{\text{P}} \circ \Phi_{\text{R}} : (o, h; g) \mapsto a. \tag{4}$$

The embodied KG is an internal component of the sLM-based policy, encapsulating the environment information.

In training, we use a fine-tuning method with soft prompts for the sLM-based policy. This is effective for adopting sLMs with limited reasoning capabilities, where in-context learning is not allowed.

Embodied KG. As the agent continuously interacts with the environment and can accumulate information for task completion, it is important to represent information efficiently for prompting the sLM-based policy. We employ an embodied KG, a set of triples $g = \{x_i = (x_i^s, x_i^r, x_i^o)\}_i$, where x^s is the subject, x^r is the relation, and x^o is the object. For instance, given "an apple is on the table" and "the agent picks up a knife", the corresponding triples are (*Apple*, *On*, *Table*) and (*Agent*, *Pickup*, *Knife*), respectively. We refine the embodied KG at each planning step t by an update function U such as

$$U : (g_{t-1},\ a_{t-1},\ o_t) \mapsto g_t. \tag{5}$$

To prompt the sLM-based policy, we also use the KG retriever function V, which retrieves a subset of triples from g relevant to observation o and task description h.

$$V : (o, h; g) \rightarrow \{x \in g | S(x, (h, o)) \geq \delta\} \tag{6}$$

The relevant triples are chosen by the pre-trained semantic relevance function S between each triple in g and inputs o and h, where δ is a threshold hyperparameter. Hereafter, g denotes the graph extracted via the KG retriever function.

Reasoning-Policy Distillation. For the reasoning-policy Φ_{R} which produces a rationale set, we employ the attention module with an encoder-decoder architecture.

$$\Phi_{\text{R}} = \Phi_{\text{Dec}} \circ \Psi \circ \Phi_{\text{Enc}} : g \mapsto \mathcal{R} \tag{7}$$

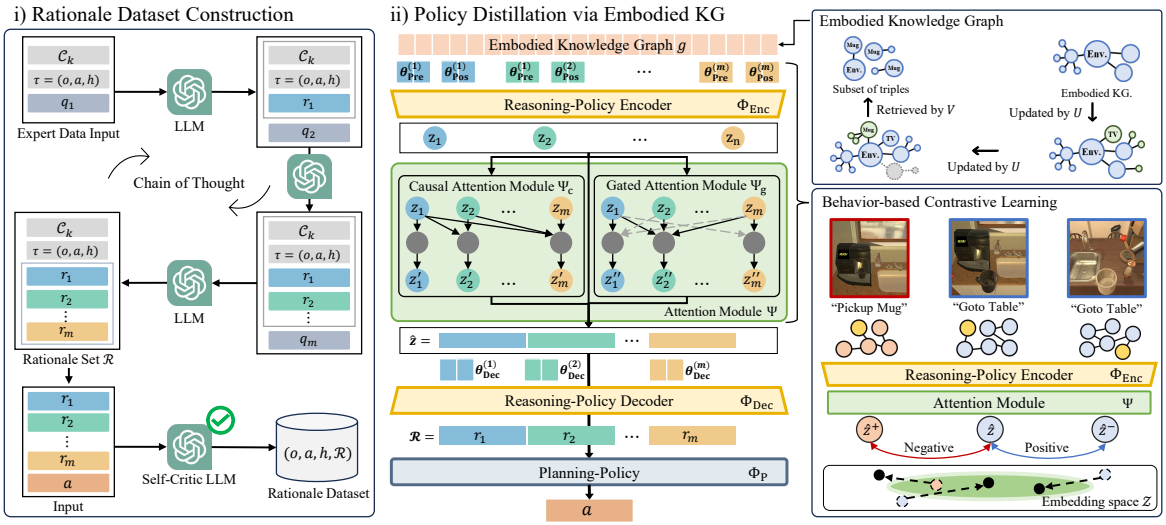


Figure 3. Distillation procedures in DEDER: During the rationale dataset construction phase, the LLM is iteratively prompted with queries q_i and rationales r_i to refine in-context examples $\mathcal{C}_k$ through retrieval augmentation. The LLM also serves as a critic, evaluating the validity of the extracted rationales $\mathcal{R}$. During the policy distillation phase, the embodied KG containing environment information as well as expert experiences is used as input g to the sLM-based reasoning-policy with the prompted casual attention, which is trained through behavior-based contrastive learning. The structure of reasoning-policy is specifically designed to produce multiple rationales $\mathcal{R}$ in a single-step CoT process through the integration of the prompted attention Ψ and the encoder-decoder architecture. The reasoning-policy is distilled from the embodied KG, which is continually updated from the dataset. Subsequently, the planning-policy Φ_{p} is trained to produce a timely action plan a, by immediately using the rationales $\mathcal{R}$ at each step.

To facilitate the single-step CoT through the reasoning-policy Φ_{R}, we also use soft prompt pools $\theta = [\theta^{(1)}, \theta^{(2)}, ..., \theta^{(m)}]$, $\theta^{(i)} \in \mathbb{R}^d$, where d is the dimension of prompt $\theta^{(i)}$ (Senadeera & Ive, 2022). The encoder Φ_{Enc} incorporates two distinct prompt pools: a prefix prompt pool θ_{Pre} and a postfix prompt pool θ_{Pos}.

$$\Phi_{\text{Enc}} : (g; \theta_{\text{Pre}}, \theta_{\text{Pos}}) \mapsto z = [z_1, ..., z_d] \tag{8}$$

Each prefix prompt $\theta^{(i)}_{\text{Pre}}$ is initialized based on the language embedding of the query q_i, while each postfix prompt $\theta^{(i)}_{\text{Pos}}$ is randomly initialized. Furthermore, for emphasizing information in each rationale and transferring it sequentially, in line with the rationale dataset construction, the attention module Ψ includes a causal attention Ψ_{c} (Vaswani et al., 2017) and gated attention Ψ_{g} (Xue et al., 2020), i.e.,

$$\hat{z} = [\hat{z}_1, ..., \hat{z}_d] = \Psi(z) = z + \alpha(\Psi_{\text{c}}(z) + \Psi_{\text{g}}(z)) \tag{9}$$

where α is a scaling factor that regulates the influence of the attention mechanisms' outputs. The decoder Φ_{Dec} utilizes a decoder prompt pool θ_{Dec} to generate a set of rationales $\mathcal{R}$.

$$\Phi_{\text{Dec}} : (\hat{z}; \theta_{\text{Dec}}) \mapsto \mathcal{R} \tag{10}$$

With the embodied KG generated by update function U and KG retriever function V from $\mathcal{D}_{\text{Rtn}}$, we optimize the reasoning-policy by the rationale reconstruction loss.

$$\mathcal{L}_{\text{Rtn}} = \mathop{\mathbb{E}}_{(o,h,\mathcal{R})\sim\mathcal{D}_{\text{Rtn}}} \left[\sum_{i=1}^{m} \log \Phi_{\text{R}}(r_i|g) \right] \tag{11}$$

This loss is calculated as the expected sum of the log-likelihoods for generating each rationale r_i.

Considering that subtle changes in the environment might lead to inconsistent agent's plan, we devise the prompted KG representations, using behavior-based contrastive learning (Stooke et al., 2021; Zhang et al., 2022; Choi et al., 2023). The prompted KG representations facilitate the causal and gated attentions for the reasoning-policy, thus enabling a single-step inference for multiple rationales. We sample a batch of embodied KG pairs $\mathcal{B}_{\text{Con}} = \{(g_i, g_i^+), (g_i, g_i^-)\}_i$, where (g_i, g_i^+) denotes a positive pair, and (g_i, g_i^-) denotes a negative pair. Specifically, the positive pair consists of embodied KG executing the same plan, while the negative pair is defined as consecutive planning steps. Then, the contrastive learning loss is formulated as

$$\mathcal{L}_{\text{Con}} = \mathop{\mathbb{E}}_{\mathcal{B}_{\text{Con}}\sim\mathcal{D}_{\text{Rtn}}} [\max\{0, d(\hat{z},\ \hat{z}^+) - d(\hat{z},\ \hat{z}^-) + \epsilon\}] \tag{12}$$

where $\hat{z} = \Psi \circ \Phi_{\text{Enc}}(g; \theta_{\text{Pre}},\ \theta_{\text{Pos}})$, d represents the sum of a distance metric within the embedding space $\hat{z} \in \mathcal{Z}$ corresponding to an element of the rationale embedding sequence (Chen et al., 2020; Oord et al., 2018), and ϵ is a margin parameter.

Planning-Policy Distillation. The planning-policy Φ_{P} predicts a next plan a based on the rationale set generated from the reasoning-policy Φ_{R}.

$$\Phi_{\text{P}} : (\mathcal{R} = \Phi_{\text{R}}(g)) \mapsto a \tag{13}$$

Algorithm 1 Policy Distillation

Rationale Dataset $\mathcal{D}_{\text{Rtn}}$
Initialize reasoning-policy Φ_{R}, and planning-policy Φ_{P}
Initialize prompt pools $\theta_{\text{Pre}}, \theta_{\text{Pos}}, \theta_{\text{Dec}}$

/ Reasoning-Policy Distillation */*
while not converge **do**
  Sample a batch $\mathcal{B} = \{(o_i, a_i, h_i)\}_i \sim \mathcal{D}_{\text{Rtn}}$
  Obtain $\mathcal{B}_{\text{Con}} = \{(g_i, g_i^+), (g_i, g_i^-)\}_i$ using (5), (6)
  Update $\Phi_{\text{R}}, \theta_{\text{Pre}}, \theta_{\text{Pos}}, \theta_{\text{Dec}}$ using (11), (12)
end while
/ Planning-Policy Distillation */*
while not converge **do**
  Sample a batch $\mathcal{B} = \{(o_i, a_i, h_i)\}_i \sim \mathcal{D}_{\text{Rtn}}$
  Obtain $\mathcal{B} = \{(a_i, g_i)\}_i$ using (5), (6)
  Calculate $\mathcal{R}$ using Φ_{R} on batch $\mathcal{B}$
  Update Φ_{P} using (14)
end while

We optimize the planning-policy via the reconstruction loss.

$$\mathcal{L}_{\text{Plan}} = \mathop{\mathbb{E}}_{(o,a,h)\sim\mathcal{D}_{\text{Rtn}}, \mathcal{R}\sim\Phi_{\text{R}}} [\log \Phi_{\text{P}}(a \mid \mathcal{R})] \tag{14}$$

Algorithm 1 lists the policy distillation procedures, where the losses in (11), (12) and the loss in (14) are used for the reasoning policy and the planning-policy, respectively.

5. Evaluation

5.1. Experiment Setting

Environments. For evaluation, we use the ALFRED (Shridhar et al., 2020) environment. For embodied reasoning tasks, ALFRED features a wide variety of interactive elements including 58 distinct object types (e.g., bread) and 26 receptacles object types (e.g., plate) across 120 different indoor scenes (e.g., kitchen). By combining these objects and indoor scenes with instructions of 7 different types (e.g., pick & place), 4703 distinct tasks can be configured (e.g., "Put a keychain in a plate and then put them in a shelf"). This setup provides a broad spectrum of real-world-like challenges, encompassing complex navigation, object manipulation, and executing sequential operations.

We use 312 trajectories for the expert dataset and organize the evaluation tasks into 4 categories based on their similarities to the tasks in the expert dataset. For **Train** category, the tasks are identical to those in the expert dataset. For **Seen** category, the tasks remain the same as those in the expert dataset, except that the starting positions of the task-irrelevant objects are placed randomly. For **Unseen Spatial** category, all objects in the environment are placed randomly. The most challenging category **Unseen Environment** includes new tasks and indoor scenes not presented in the expert dataset. The environment details are in Appendix A.

Baselines. For comparison, we implement several language planning approaches: 1) **SayCan** (Brohan et al., 2023) is an embodied planning framework that integrates the probability from an LLM with affordance scores. For embodied control, the affordance is based on object presence information. 2) **ZSP** (Huang et al., 2022) employs a step-wise planning to accomplish the embodied tasks. 3) **LLM-planner** (Song et al., 2023), directly utilizes an LLM for embodied task planning, which dynamically re-plans when it fails to generate an executable plan. In evaluating in off-the-shelf devices, we adopt sLMs for these language planning baselines (SayCan, ZSP, LLM-planner).

We also implement several knowledge distillation algorithms: 4) **SCoTD** (Li et al., 2023) is a knowledge distillation algorithm to train an sLM using reasoning samples derived from an LLM. 5) **SCOTT** (Wang et al., 2023) is a knowledge distillation method to train an sLM, which involves self-consistent CoT augmentation from an LLM and counterfactual reasoning objectives. 6) **End2End** (Micheli & Fleuret, 2021) is an embodied task planning method using a single-tier policy unlike DEDER, which directly generates a plan from the inputs. To evaluate the task planning performance in the environment through generated trajectories, we also implement an additional rule-based policy that directly interacts with the environment, following the action plans from the baselines and our DEDER.

Evaluation metrics. We use two different metrics in ALFRED (Shridhar et al., 2020). **Task Success Rate (SR)** (%) is the percentage of tasks fully completed, where a task is regarded as a success if and only if all the sub-goals are achieved. For example, the task "Slice a heated bread" is decomposed into individual sub-goals like "slice the bread" and "heat the bread". **Goal-conditioned Success Rate (GC)** (%) is the percentage of sub-goals that are completed.

5.2. Performance Evaluation

In Table 1, we evaluate the embodied task planning performance, wherein each policy is evaluated in a zero-shot manner. Our DEDER consistently demonstrates the robust performance in both SR and GC metrics across all test categories (Train, Seen, Unseen Spatial, Unseen Environment), achieving 21.6% higher SR and 12.3% higher GC on average over the most competitive baseline LLM-planner-PaLM. Given that LLM-planner-PaLM exploits the PaLM (Chowdhery et al., 2023) with 540 billion parameters, 2700 times larger than DEDER, this performance gain of DEDER is particularly significant. Moreover, compared to the baselines that have the same parameter size, we observe that DEDER outperforms these baselines for all categories up to 27.6% higher SR and 12.6% higher GC on average.

The language planning baselines (SayCan, LLM-Planner, ZSP), which are configured to adopt sLMs (LLaMA2,

Table 1. Performance of embodied task planning in ALFRED with 4 different task categories

Method	Train		Seen		Unseen Spatial		Unseen Environment	
	SR	GC	SR	GC	SR	GC	SR	GC
LLM-based policy: PaLM (540B), LLaMA2 (7B)								
SayCan-PaLM	34.1±0.0	68.8±0.0	37.6±0.0	66.8±0.0	26.5±0.0	66.5±0.0	29.3±0.0	**68.8±0.0**
LLM-planner-PaLM	70.9±0.0	86.6±0.0	66.8±0.0	84.3±0.0	33.6±0.0	67.6±0.0	17.2±0.0	54.3±0.0
ZSP-PaLM	73.8±0.0	89.2±0.0	59.6±0.0	80.7±0.0	28.8±0.0	66.5±0.0	6.9±0.0	36.5±0.0
SayCan-LLaMA2	0.0±0.0	10.6±0.0	0.3±0.0	9.6±0.0	0.0±0.0	10.3±0.0	0.0±0.0	2.2±0.0
LLM-planner-LLaMA2	1.8±0.0	19.6±0.0	2.0±0.0	22.7±0.0	0.8±0.0	19.6±0.0	0.0±0.0	15.8±0.0
ZSP-LLaMA2	54.3±0.0	76.5±0.0	26.7±0.0	59.9±0.0	6.7±0.0	46.9±0.0	0.0±0.0	26.6±0.0
sLM-based policy: GPT2-large (0.8B), GPT2 (0.2B)								
SayCan-GPT2-large	0.2±0.0	14.7±0.0	0.5±0.0	17.1±0.0	0.5±0.0	17.6±0.0	0.0±0.0	18.1±0.0
LLM-planner-GPT2-large	0.0±0.0	3.43±0.0	0.0±0.0	4.0±0.0	0.0±0.0	2.0±0.0	0.0±0.0	1.8±0.0
ZSP-GPT2-large	1.8±0.0	3.6±0.0	0.8±0.0	3.4±0.0	0.3±0.0	4.3±0.0	0.0±0.0	0.4±0.0
End2End-GPT2-large	41.1±12.2	63.5±11.6	25.2±7.0	54.3±10.9	11.4±4.5	50.1±9.2	5.7±1.0	53.8±25.3
SCoTD-GPT2	55.8±4.2	82.7±1.5	51.8±5.2	79.0±2.3	29.3±2.1	70.4±0.9	27.6±1.7	59.8±1.7
SCOTT-GPT2	62.2±1.6	85.6±0.1	57.2±4.0	81.3±1.6	32.7±2.1	72.0 ± 0.1	24.1 ± 7.9	60.3±1.6
End2End-GPT2	33.1±4.6	46.6±8.1	17.6±2.6	38.8±8.0	8.5±6.3	36.3±9.1	3.4±0.9	34.6±9.0
DEDER-GPT2	**100.0±0.0**	**100.0±0.0**	**81.8±0.5**	**92.2±0.2**	**52.7±1.0**	**81.2±0.4**	**40.3±0.9**	68.7±0.6

GPT2-large), exhibit low performance. This is due to sLMs' limited reasoning capabilities. Meanwhile, the knowledge distillation baselines (SCoTD, SCOTT) maintain decent performance. While they use distillation from an LLM via few-shot prompting, their distilled knowledge is somewhat limited by the conventional CoT mechanism. This limitation arises because they do not employ multi-step prompting and self-verification, unlike DEDER. Furthermore, the End2End baseline exhibits significantly low performance in directly conducting embodied task planning with the expert dataset, due to the limited reasoning capability of the sLM.

In contrast, our framework employs the rationale dataset and the sLM-based policy with a two-tier hierarchy structure. This enables the effective distillation of the LLM's reasoning capabilities, specifically tailored for embodied task planning based on MDP-featured in-context learning.

5.3. Ablation Studies

In the ablation studies, the performance metrics for all test categories (Train, Seen, Unseen) are reported in SR.

Rationale Dataset Construction. For extracting rationales and constructing the dataset in Section 4.1, we test several language models, including sLMs such as GPT2-large (Radford et al., 2019) denoted as GPT2, and LLMs such as PaLM and GPT3 (Chowdhery et al., 2023; Brown et al., 2020). We also evaluate the dataset construction process without employing MDP-featured in-context learning and self-verification; This ablated method is denoted as *Few-shot*, as described in (Wei et al., 2022), where a fixed set of examples is used for prompting rationale extraction.

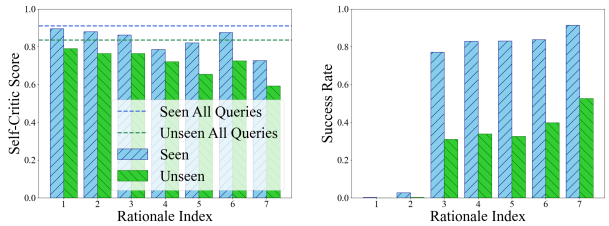
In Table 2, there is a notable performance drop across the task categories when employing GPT2. These results are consistent with our motivation to harness the reasoning capabilities of LLMs for rationale extraction, which in turn contributes to the effective distillation into the sLM-based policy. Moreover, DEDER yields better performance compared to *Few-shot* by an average of 5.35% in the Unseen settings, excluding GPT2 results. This improvement indicates the benefits of our MDP-featured in-context learning and self-verification methods.

Table 2. Ablation on rationale dataset construction

Method	LM	Train	Seen	Unseen
Few-shot	GPT2	41.9±19.3	2.4±0.5	0.3±0.1
DEDER	GPT2	60.8±4.0	53.5±2.2	23.4±8.2
Few-shot	GPT3	100.0±0.0	72.8±0.1	38.6±1.5
DEDER	GPT3	100.0±0.0	72.8±0.1	42.2±1.2
Few-shot	PaLM	100.0±0.0	76.9±0.3	39.6±0.1
DEDER	PaLM	100.0±0.0	**81.8±0.5**	**46.5±1.0**

Rationale Structure. We analyze the effect of individual queries designed for rationale extraction. In Figure 4(a), we evaluate the rationale set generated by the reasoning-policy involving the LLM's self-critic function in (3). The dotted line denotes the performance achieved by employing all 7 queries, whereas each bar along the x-axis indicates the performance when the i-th query is excluded during the dataset construction. Since each query is specifically designed to capture unique features in MDPs, such as goals, state, and return-to-go (illustrated in Figure 2), the exclusion of any one of these queries leads to a performance decline. In Figure 4(b), each bar along the x-axis represents the performance achieved when the rationale set is formulated

with the queries up to the i-th. In the rationale generation process, the 1st and 2nd queries encapsulate general information that is applicable to any of the tasks. From the 3rd query onwards, the reasoning becomes increasingly specific to a given task. Thus, we observe the best performance when all queries are used, building upon the comprehensive information generated in the earlier steps.



(a) Rationale omission (b) Incomplete rationales

Figure 4. Ablation on each rationale

Reasoning-policy structure. Table 3 shows the effect of our embodied KG and contrastive learning scheme $\mathcal{L}_{\text{Con}}$. The results indicate that DEDER, when utilizing both KG and $\mathcal{L}_{\text{Con}}$, achieves the highest performance. This is attributed to our embodied KG, which efficiently encapsulates both the evolving embodied information and the agent's interaction experiences. Additionally, in the absence of contrastive learning, the reasoning-policy struggles to extract precise features for the next plan, leading to a performance drop. We also measure the inference time of DEDER and DEDER without embodied KG on off-the-shelf devices such as RTX 3090 and 3050 GPUs. As the use of embodied KG allows for more efficient representation, DEDER achieves a reduction in inference time by 0.3 second on average.

Table 3. Ablation on embodied KG and contrastive learning

Method	Inference Time		Seen	Unseen
	RTX 3090	RTX 3050		
DEDER	0.65±0.01	1.16±0.01	**81.8±0.5**	**46.5±1.0**
– $\mathcal{L}_{\text{Con}}$	-	-	77.2±0.5	42.5±1.5
– KG	0.72±0.01	1.68±0.01	71.0±6.3	42.0±6.4
– KG & $\mathcal{L}_{\text{Con}}$	-	-	73.2±0.5	43.3±0.8

Table 4 shows the effect of our attention structure Ψ consisting Ψ_{c} and Ψ_{g}, which are used in the reasoning-policy Φ_{R}. In the table, *Iterative* specifies that Φ_{R} is inferred m times sequentially to generate the rationale set without using the attention mechanism. Ψ_{a} denotes the basic attention structure (Vaswani et al., 2017). Considering both the success rate and inference time, DEDER not only efficiently distills rationales but also offers an effective inference framework for off-the-shelf agents. This is significant when considering that LLM-planner-LLaMA2 reaches 9.37 seconds at maximum resource usage on an RTX 3090, as in Table 1.

Table 4. Ablation on attention structure in reasoning-policy

Method	Inference Time		Seen	Unseen
	RTX 3090	RTX 3050		
Iterative	2.92±0.02	3.08±0.01	76.0±0.7	44.6±1.2
DEDER	0.65±0.01	1.16±0.01	**81.8±0.5**	**46.5±1.0**
– Ψ_{c}	0.63±0.01	1.16±0.01	75.9±0.3	40.6±0.6
– Ψ_{g}	0.63±0.01	1.16±0.01	76.8±0.3	40.7±1.4
– Ψ_{c} & Ψ_{g}	0.60±0.01	1.00±0.01	63.3±0.1	34.1±0.1
DEDER w Ψ_{a}	0.63±0.01	1.17±0.01	74.7±0.4	43.6±0.1

sLM Capacity. Table 5 shows the performance of DEDER with respect to the variations in network parameter sizes for the reasoning-policy Φ_{R} and the planning-policy Φ_{P}. In our default framework implementation, we utilize the t5-small and gpt2 models for Φ_{R} and Φ_{D}, respectively. The results indicate that the performance improvement is not significant when the parameter size of the planning-policy Φ_{P} increases. In contrast, enhancing the parameter size of the reasoning-policy Φ_{R} results in performance gains, showing an average increase of 6.57% when comparing the t5-small and the t5-large used for Φ_{R} in unseen settings. Specifically, the smaller sLM (t5-small) tends to overfit on the training datasets, which might yield better performance in the Seen category compared to the mid-sized sLM (t5-base). For the larger sLM (t5-large), a performance improvement is noted in the Seen category, attributed to its enhanced reasoning capabilities. In contrast, the Unseen settings demonstrate a linear performance increase as the parameter size of the reasoning policy grows, suggesting that a larger parameter size significantly boosts the generalization ability of the model. This indicates the benefits of distilling rationales from LLMs, which plays a crucial role in establishing a robust sLM-based policy.

Table 5. DEDER performance w.r.t. policy network sizes

Φ_{R}	Φ_{P}	Parameter Size	Seen	Unseen
t5-small	gpt2	0.06B+0.1B	81.8±0.5	46.5±1.0
t5-small	gpt2-medium	0.06B+0.4B	81.4±0.1	46.1±0.5
t5-small	gpt2-large	0.06B+0.8B	81.8±0.1	46.1±1.1
t5-base	gpt2	0.2B+0.1B	79.4±0.5	48.5±1.2
t5-base	gpt2-medium	0.2B+0.4B	78.8±0.6	48.6±0.6
t5-base	gpt2-large	0.2B+0.8B	80.0±1.3	49.0±1.5
t5-large	gpt2	0.7B+0.1B	82.1±1.0	52.5±0.7
t5-large	gpt2-medium	0.7B+0.4B	81.8±0.6	52.9±0.9
t5-large	gpt2-large	0.7B+0.8B	81.2±0.4	53.3±0.7

6. Conclusion

We introduced DEDER, a novel framework that effectively distills the reasoning capabilities of LLMs into more compact sLMs for executing complex embodied tasks in device-constrained environments. The framework operates in a strategic distillation process, involving embodied rational

data construction from an LLM, data-driven embodied policy distillation to an sLM, and task planning with the sLM. This allows for the efficient use of LLM-powered complex task planning functions in real-world time-constrained settings while ensuring the adaptability to resource-constrained agent conditions through two-step distillation into reasoning and decision-making.

Limitation. As DEDER employs pre-trained sLMs, there is a potential dependency on the pre-trained knowledge embedded in the sLMs. In Table 5, we observe that a reduced network capacity of sLMs leads to decreased performance in unseen settings. This indicates that the limited network capacity of the sLM hinders the distillation of reasoning capabilities, consequently affecting the zero-shot adaptation in environments with significant domain shifts.

Future Work. Future directions for our research include enhancing the framework's ability for few-shot optimization, especially in scenarios with significant domain shifts, aiming to explore the versatility of LLMs.

Acknowledgements

We would like to thank anonymous reviewers for their valuable comments and suggestions. This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2022-0-01045, No. 2022-0-00043, No. 2019-0-00421, No. 2020-0-01821), by the National Research Foundation of Korea (NRF) grant funded by MSIT (No. RS-2023-00213118), and by Samsung Electronics.

Impact Statement

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

References

Aeronautiques, C., Howe, A., Knoblock, C., McDermott, I. D., Ram, A., Veloso, M., Weld, D., SRI, D. W., Barrett, A., Christianson, D., et al. Pddl: The planning domain definition language. *Technical Report, Tech. Rep.*, 1998.

Andrus, B. R., Nasiri, Y., Cui, S., Cullen, B., and Fulda, N. Enhanced story comprehension for large language models through dynamic document-based knowledge graphs. In *Proceedings of the 36th Conference on Artificial Intelligence*, pp. 10436–10444, 2022.

Baek, J., Aji, A. F., and Saffari, A. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. *arXiv preprint arXiv:2306.04136*, 2023.

Brohan, A., Chebotar, Y., Finn, C., Hausman, K., Herzog, A., Ho, D., Ibarz, J., Irpan, A., Jang, E., Julian, R., et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Proceedings of the 6th Conference on Robot Learning*, pp. 287–318, 2023.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. In *Proceedings of the 33rd Advances in Neural Information Processing Systems*, 2020.

Chane-Sane, E., Schmid, C., and Laptev, I. Goal-conditioned reinforcement learning with imagined subgoals. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 1430–1440, 2021.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. In *Proceedings of the 35th Advances in Neural Information Processing Systems*, pp. 15084–15097, 2021.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. E. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 1597–1607, 2020.

Choi, W., Kim, W. K., Kim, S., and Woo, H. Efficient policy adaptation with contrastive prompt ensemble for embodied agents. In *Proceedings of 37th Advances in Neural Information Processing Systems*, 2023.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

Dasgupta, I., Kaeser-Chen, C., Marino, K., Ahuja, A., Babayan, S., Hill, F., and Fergus, R. Collaborating with language models for embodied reasoning. *arXiv preprint arXiv:2302.00763*, 2023.

Driess, D., Xia, F., Sajjadi, M. S. M., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., Huang, W., Chebotar, Y., Sermanet, P., Duckworth, D., Levine, S., Vanhoucke, V., Hausman, K., Toussaint, M., Greff, K., Zeng, A., Mordatch, I., and Florence, P. Palm-e: An embodied multimodal language model. In *arXiv preprint arXiv:2303.03378*, 2023.

Hausknecht, M. J. and Stone, P. Deep recurrent q-learning for partially observable mdps. In *AAAI Fall Symposium*, pp. 29–37, 2015.

Huang, W., Abbeel, P., Pathak, D., and Mordatch, I. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 9118–9147. PMLR, 2022.

Jain, U., Liu, I.-J., Lazebnik, S., Kembhavi, A., Weihs, L., and Schwing, A. G. Gridtopix: Training embodied agents with minimal supervision. In *Proceedings of the 18th IEEE/CVF International Conference on Computer Vision*, pp. 15141–15151, 2021.

Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. In *Proceedings of the 35th Advances in Neural Information Processing Systems*, pp. 1273–1286, 2021.

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense passage retrieval for open-domain question answering. In *Proceedings of the 15th Conference on Empirical Methods in Natural Language Processing*, pp. 6769–6781, 2020.

Kullback, S. and Leibler, R. A. On information and sufficiency. *The Annals of Mathematical Statistics*, 1951.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Proceedings of the 34th Advances in Neural Information Processing Systems*, 2020.

Li, L. H., Hessel, J., Yu, Y., Ren, X., Chang, K.-W., and Choi, Y. Symbolic chain-of-thought distillation: Small models can also "think" step-by-step. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pp. 2665–2679, 2023.

Micheli, V. and Fleuret, F. Language models are few-shot butlers. *arXiv preprint arXiv:2104.07972*, 2021.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Parisotto, E., Ba, J. L., and Salakhutdinov, R. Actor-mimic: Deep multitask and transfer reinforcement learning. In *Proceddings of the 4th International Conference on Learning Representations*, 2016.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21 (140):1–67, 2020.

Ram, O., Levine, Y., Dalmedigos, I., Muhlgay, D., Shashua, A., Leyton-Brown, K., and Shoham, Y. In-context retrieval-augmented language models. *arXiv preprint arXiv:2302.00083*, 2023.

Schmitt, S., Hudson, J. J., Zidek, A., Osindero, S., Doersch, C., Czarnecki, W. M., Leibo, J. Z., Kuttler, H., Zisserman, A., Simonyan, K., et al. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*, 2018.

Senadeera, D. C. and Ive, J. Controlled text generation using t5 based encoder-decoder soft prompt tuning and analysis of the utility of generated text in ai. *arXiv preprint arXiv:2212.02924*, 2022.

Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., and Fox, D. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the 31st IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10737–10746, 2020.

Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., and Garg, A. Progprompt: Generating situated robot task plans using large language models. In *Proceedings of the 40th IEEE International Conference on Robotics and Automation*, pp. 11523–11530, 2023.

Song, C. H., Wu, J., Washington, C., Sadler, B. M., Chao, W.-L., and Su, Y. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the 19th IEEE/CVF International Conference on Computer Vision*, pp. 2998–3009, 2023.

Stooke, A., Lee, K., Abbeel, P., and Laskin, M. Decoupling representation learning from reinforcement learning. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 9870—-9879, 2021.

Sumers, T., Marino, K., Ahuja, A., Fergus, R., and Dasgupta, I. Distilling internet-scale vision-language models into embodied agents. *arXiv preprint arXiv:2301.12507*, 2023.

Sun, J., Luo, Y., Gong, Y., Lin, C., Shen, Y., Guo, J., and Duan, N. Enhancing chain-of-thoughts prompting with iterative bootstrapping in large language models. *arXiv preprint arXiv:2304.11657*, 2023.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT Press, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Proceedings of the 31st Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Wang, P., Wang, Z., Li, Z., Gao, Y., Yin, B., and Ren, X. Scott: Self-consistent chain-of-thought distillation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pp. 5546–5558, 2023.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th Advances in Neural Information Processing Systems*, 2022.

Wu, Z., Wang, Z., Xu, X., Lu, J., and Yan, H. Embodied task planning with large language models. *arXiv preprint arXiv:2307.01848*, 2023.

Xue, L., Li, X., and Zhang, N. L. Not all attention is needed: Gated attention network for sequence data. In *Proceedings of the 34th Conference on Artificial Intelligence*, pp. 6550–6557, 2020.

Yin, H. and Pan, S. Knowledge transfer for deep reinforcement learning with hierarchical experience replay. In *Proceedings of the 31st Conference on Artificial Intelligence*, pp. 1640–1646, 2017.

Zhang, Q., Peng, Z., and Zhou, B. Learning to drive by watching youtube videos: Action-conditioned contrastive policy pretraining. In *Proceedings of the 17th European Conference on Computer Vision*, pp. 111–128, 2022.

A. Environment settings

A.1. ALFRED

We utilize ALFRED (Shridhar et al., 2020), which provides comprehensive vision-and-language navigation and rearrangement tasks for embodied AI. This environment requires an agent to follow language formatted instructions to accomplish real-world-like household tasks. ALFRED features 58 different object types (e.g., bread) and 26 receptacle types (e.g., plate) across 120 various indoor scenes (e.g., kitchen). It supports 4703 unique tasks, each configured by combining these elements with one of 7 instruction types (e.g., pick & place), such as "Put a keychain in a plate and then put them on a shelf". This complexity and diversity makes ALFREDD an ideal benchmark for evaluating models that emphasize hierarchy, modularity, and advanced reasoning and planning capabilities. The detail of instructions and excutable plans are listed in Table A.6. Furthermore, the visualizations of various indoor scenes and observations in ALFRED are shown in Figure A.1

Table A.6. Instructions and executable plans in ALFRED environment

	Type	**Example**
Instructions	Pick & Place	Put a watch on a table.
	Stack & Place	Put a bowl with a spoon in it on the table.
	Pick Two & Place	Put two pencils in a drawer.
	Clean & Place	Put a clean rag on the top shelf of a barred rack.
	Heat & Place	Put a cooked potato slice on the counter
	Cool & Place	Put a slice of cold lettuce on a counter.
	Examine & in Light	Pick up a book and turn on a lamp.
Plans	OpenObject [Object]	OpenObject GarbageCan
	CloseObject [Object]	CloseObject GarbageCan
	ToggleObject [Object]	ToggleObject FloorLamp
	SliceObject [Object]	SliceObject Potato
	GotoLocation [Receptacle Object]	GotoLocation SideTable
	PickupObject [Object] [Receptacle Object]	PickupObject ButterKnife SideTable
	PutObject [Object] [Receptacle Object]	PutObject Pan DiningTable
	CoolObject [Object] [Receptacle Object]	CoolObject Apple Fridge
	HeatObject [Object] [Receptacle Object]	HeatObject Mug Microwave
	CleanObject [Object] [Receptacle Object]	CleanObject Tomato Sink
	End	End

A.2. Expert Dataset

Expert Dataset and Evaluation Task Settings. To generate an expert dataset, we use planning domain definition language rules (Aeronautiques et al., 1998). For implementation, we use the open source project[1]. We collect 312 expert trajectories in a variety of tasks varying the starting positions of the agent and objects as well as the indoor scenes.

We organize the evaluation tasks into 4 distinct categories based on task similarities with the expert dataset: **Train**, **Seen**, **Unseen Spatial**, and **Unseen Environment**. For the **Train** category, the tasks are identical to those tasks in the expert dataset. The **Seen** category maintains the same tasks as in the expert dataset, but task-irrelevant objects are randomly positioned at the start. For this, we evaluate 528 tasks. In the **Unseen Spatial** category, all objects are placed randomly, and tasks are either defined by new task descriptions or optimal planning sequences not included in the Train category. For this, we evaluate 1415 tasks. Lastly, for the most challenging category **Unseen Environment** where all objects are randomly placed, and the task or indoor scenes are not presented in the Train category. We utilize 58 tasks for evaluating the Unseen Environment category. For all models that require training, we conduct evaluations using three distinct seeds and report their average performance, along with the associated variances.

B. Implementation Details

In this section, we provide the implementation details of our proposed framework DEDER and each comparison. Our framework is implemented using Python v3.9 and PyTorch v2.0.1, trained on a system of an Intel(R) Core (TM) i9-10980XE processor and an NVIDIA RTX A6000 GPU. For comparisons, we implement 3 types of widely used approaches: language

[1]https://github.com/askforalfred/alfred

(a) Example of Heat & Place task.

(b) Example of Pick Two & Place task.

(c) Example of Examine & in Light task.

(d) Example of Pick & Place task.

Figure A.5. Task examples set within different indoor scenes. The observation includes a variety of objects with which the agent can interact and alter states to complete the given task.

planning, knowledge distillation, and end-to-end methodologies.

B.1. Language Planning Approach

For the language planning approaches, we employ 3 different methodologies: SayCan, LLM-planner, and ZSP (Brohan et al., 2023; Song et al., 2023; Huang et al., 2022). For generating high-level plans, we utilize various LMs like PaLM, LLAMA, and GPT2-large.

SayCan (Brohan et al., 2023) integrates pretrained skills with language models, generating plans that are feasible to the context. SayCan achieves this by combining affordance scores derived from the LM with the agent's experiences. In line with SayCan's methodology, we calculate embodied affordance scores by utilizing object presence information. For implementation, we refer to the open source project[2].

ZSP (Huang et al., 2022) leverages the capabilities of the LLMs for embodied task planning by interpreting high-level task descriptions and formulating sequential strategies, thus efficiently performing embodied tasks. ZSP accomplishes this by crafting step-by-step prompts based on examples of similar successful tasks, followed by sampling executable plans using the LLM in conjunction with these provided examples. For implementation, we refer to the open source project[3].

LLM-planner (Song et al., 2023) leverages the LLMs for few-shot planning, empowering embodied agents to perform complex tasks in environments with observed information, guided by natural language instructions. For implementation, we

[2]https://github.com/google-research/google-research/tree/master/saycan

[3]https://github.com/huangwl18/language-planner

refer to the open source[4].

In text generation configuration, temperature controls the degree of randomness in the generation process. A lower temperature results in more predictable and consistent text, while a higher temperature can produce more diverse and sometimes unexpected outcomes. Top k sampling limits the model to consider only the top k most probable next words when choosing the next word in the sequence. This method helps to constrain randomness, thereby enhancing the quality of the generated text. Top p involves selecting the smallest set of words whose cumulative probability exceeds p for choosing the next word.

The hyperparameter settings for language planning approaches are summarized in Table A.7.

Table A.7. Hyperparamer settings for language planning approaches

Hyperparameters	**Value**
LLM Configuration	
PaLM	text-bison-001
LLaMA2	llama-2-7b
GPT2	gpt2-large
In-Context Example Retriever	paraphrase-MiniLM-L6-v2 (LLM-Planner) stsb-roberta-large (ZSP)
Number of Prompts	4 (LLM-Planner, ZSP)
Text Generation Configuration	
Sampling Method	beam search
Beam Size	3
Temperature	0.01
Top k	5
Top p	0.3
Maximum New Tokens	40

B.2. Knowledge Distillation Approach

For the knowledge distillation approaches, we employ two different algorithms: SCoTD and SCOTT (Li et al., 2023; Wang et al., 2023). For distilling the reasoning-policy to produce MDP-featured rationales, we implement each method to create the rationale dataset accordingly.

SCoTD (Li et al., 2023) is a CoT distillation method to train an sLM. It utilizes a LLM to generate a variety of rationales with answers, which are then used to educate the sLM. We employ SCoTD to generate and train rationale data for the reasoning-policy, and then utilize the distilled rationales to further train the planning-policy.

SCOTT (Wang et al., 2023) is a consistency knowledge distillation method to train a smaller, self-consistent CoT model from a much larger teacher model. SCOTT uses contrastive decoding to elicit better rationale supervision and a counterfactual reasoning objective to align the student model's predictions with these rationales. We utilize SCOTT for creating rationale data and subsequently training the reasoning-policy. The learned rationales from reasoning-policy are then applied to train the planning-policy. For implementation, we refer to the open source project[5].

The hyperparameter settings for knowledge distillation approaches are summarized in Table A.8.

[4]https://github.com/OSU-NLP-Group/LLM-Planner/

[5]https://github.com/wangpf3/consistent-CoT-distillation

Table A.8. Hyperparamer settings for knowledge distllation approaches

Hyperparameters	Value
Source LLM	
PaLM	text-bison-001
Temperature	0.7 (SCoTD), 0.1 (SCOTT)
Return samples	3 (SCoTD), 1 (SCOTT)
Reasoning-policy	
sLM	t5-small
Train epochs	100
Batch size	1
Optimizer	SGD
Learning rate	5e−5
Planning-policy	
sLM	gpt2
Train epochs	20
Batch size	2
Optimizer	SGD
Learning rate	3e−5
Text Generation Configuration	
Sampling Method	beam search
Beam Size	3
Temperature	0.01
Top k	5
Top p	0.3
Maximum New Tokens	40

B.3. End2End

End2End (Micheli & Fleuret, 2021) is a method for embodied task planning that specifically utilizes the GPT-2 model, trained with direct supervision on expert data. This approach forms the foundational backbone for our planning policy Φ_{P} implementation. For implementation, we refer to the open source project[6]. The hyperparameter settings for End2End are summarized in Table A.9.

Table A.9. Hyperparamer settings for End2End

Hyperparameters	Value
sLM	gpt2
Train epochs	100
Batch size	1
Optimizer	SGD
Learning rate	3e−5

B.4. DEDER

The entire procedure of our DEDER consists of rationale dataset construction and policy distillation via embodied knowledge graph phases.

[6]https://github.com/vmicheli/lm-butlers

Algorithm 2 Rationale Dataset Construction

Expert Dataset $\mathcal{D}_{\text{exp}} = \{\tau_i\}_i$, LLM Φ_{LLM}

Initialize rationale dataset, $\mathcal{D}_{\text{Rtn}} \leftarrow \emptyset$
for $\tau_i = (o, a, h)$ in $\mathcal{D}_{\text{exp}}$ **do**
  while true **do**
    Sample a set of tuples, $\mathcal{C} = \{c_1, ..., c_n\} \sim \mathcal{D}_{\text{Rtn}}$
    Retrieve in-context example set, $\mathcal{C}_k = F(\tau, \mathcal{C})$
    Initialize rationale set, $\mathcal{R} \leftarrow \emptyset$
    for $j = 1, ..., m$ **do**
      Generate rationale r_j given query q_j and (2)
      Update $\mathcal{R} \leftarrow \mathcal{R} \cup \{r_j\}$
    end for
    Construct new tuple $c = (o, a, h, \mathcal{R})$
    if c passes the self-critic using (3) **then**
      $\mathcal{D}_{\text{Rtn}} \leftarrow \mathcal{D}_{\text{Rtn}} \cup \{c\}$
      break
    end if
  end while
end for
return $\mathcal{D}_{\text{Rtn}} = \{c_i\}_i$

B.4.1. Rationale Dataset Construction

In the rationale dataset construction phase, we use PaLM (Chowdhery et al., 2023) as the source LLM, exploiting its reasoning capabilities. We formulate 7 queries to extract rationales from the LLM and manually design 9 initial examples of query-rationale pairs for each expert transition. To calculate similarity between language embeddings of τ and c, we use contextual embedding model, To utilize the LLM as a critic function, we query the LLM to assess whether the generated rationales are sufficient to generate the plan. For instance, we ask, 'Can the rationale {*rationale*} lead to the next plan {*plan*}? Answer with yes or no.'. To ensure accurate evaluations, we pose several variations on the critic prompt and determine the final critic score based on majority voting.

Algorithm 2 lists the dataset construction procedures.

B.4.2. Policy Distillation via Embodied Knowledge Graph

In the policy distillation phase, we distill the reasoning capabilities from the LLM into an sLM-based policy Φ_{sLM}, which is structured with a two-tier hierarchy consisting of the reasoning-policy Φ_{R} and the planning-policy Φ_{P}.

Reasoning-policy. For the reasoning-policy Φ_{R}, we utilize a pre-trained language model with an encoder-decoder structure, specifically t5-small (Raffel et al., 2020), as our default setting. The dimension of prefix prompts $\theta^{(i)}_{\text{Pre}}$, postfix prompts $\theta^{(i)}_{\text{Pos}}$ and decoder prompts $\theta^{(i)}_{\text{Dec}}$ are set to be 20. Our implementation of the attention module Ψ incorporates two distinct attention mechanisms: causal attention and gated attention, each comprising a single attention layer. The causal attention module uses a causal mask, while the gated attention module includes an additional learnable gate function. Φ_{R} is optimize by (11) and (12).

Planning-policy. For the planning-policy Φ_{P}, we utilize a pre-trained language model with a decoder structure, specifically gpt2 (Radford et al., 2019), as our default setting. Φ_{P} is optimize by (14).

The hyperparameter settings are summarized in Table A.10.

C. Additional Experiments

For further investigation, we report additional experimental results.

Table A.10. Hyperparamer settings for DEDER

Hyperparameters	Value	Hyperparameters	Value
Source LLM		**Planning-policy**	
PaLM	text-bison-001	sLM	gpt2
Temperature	0.1	Train epochs	20
Return samples	1	Batch size	2
Reasoning-policy		Optimizer	SGD
sLM	t5-small	Learning rate	3e−5
Encoder prompt length	20 ($\theta_{\text{Pre}}^{(i)}$), 20 ($\theta_{\text{Pos}}^{(i)}$)	**Text Generation Configuration**	
Decoder prompt length	20 ($\theta_{\text{Dec}}^{(i)}$)	Sampling Method	beam search
Train epochs	100	Beam Size	3
Batch size	1	Temperature	0.01
Optimizer	SGD	Top k	5
Learning rate	5e−5	Top p	0.3
Scaling factor α	0.5	Maximum New Tokens	40

C.1. Details of Ablation on Rationale Dataset Construction

Table A.11 shows detailed experiment results of Table 2. DEDER consistently demonstrates improved performance than the few-shot CoT method across various language models. The lower performance is observed when using GPT2 for rationale extraction, demonstrating the limited reasoning capability of sLM with in-context learning. The slight performance difference between the Few-shot approach and DEDER, when using GPT3.5's chat-based architecture, can be attributed to its conversational design focus. For the optimal application of MDP-featured in-context learning with a Chat LLM, distinct from a text generation model, crafting an in-context example design specifically for dialogue interactions becomes crucial.

Table A.11. Details of ablation on rationale dataset construction

Method	LM	Train		Seen		Unseen Spatial		Unseen Environment	
		SR	GC	SR	GC	SR	GC	SR	GC
Few-shot	GPT2	41.9±19.3	67.4±13.5	2.4±0.5	23.1±0.7	0.6 ± 0.1	22.1±0.9	0.0±0.0	20.5 ± 1.2
DEDER	GPT2	60.8±4.0	82.4±2.2	53.5±2.2	75.3±1.3	27.3±61.7	61.7±0.7	19.4±15.5	49.1±2.8
Few-shot	GPT3	100.0±0.0	100.0±0.0	72.8±0.1	87.4±0.2	48.6±0.4	79.0±0.1	28.7±2.6	62.9±2.1
DEDER	GPT3	100.0±0.0	100.0±0.0	72.8±0.1	88.8±0.2	46.9±0.5	77.3±0.4	37.4±2.0	65.3±0.9
Few-shot	GPT3.5	100.0±0.0	100.0 ± 0.0	78.3±0.6	90.5±0.2	52.0±0.7	80.0±0.3	39.7±0.0	66.5±0.3
DEDER	GPT3.5	100.0±0.0	100.0±0.0	78.4±1.4	91.5±2.3	50.0±0.6	80.3±0.2	39.9±2.3	68.6±1.4
Few-shot	PaLM	100.0±0.0	100.0 ± 0.0	76.9±0.3	89.7±0.1	49.3±0.3	79.7±0.2	29.9±1.0	63.5±0.4
DEDER	PaLM	100.0±0.0	100.0±0.0	**81.8±0.5**	**92.2±0.2**	**52.7±1.0**	**81.2±0.4**	**40.3±0.9**	**68.7±0.6**

C.2. Details of Embodied Knowledge Graph and Contrastive Learning

Table A.12 shows detailed experiment results of Table 3. We measured inference times several off-the-shelf devices such as RTX 3090, 3050 and 2080 Ti GPUs. DEDER ensures real-time inference speeds across these various devices while consistently yielding superior performance compared to other ablated comparisons.

Table A.12. Details of ablation on embodied KG and contrastive learning

Method	Inference Time			Train		Seen		Unseen Spatial		Unseen Environment	
	RTX 3090	RTX 3050	RTX 2080 Ti	SR	GC	SR	GC	SR	GC	SR	GC
DEDER	0.65±0.01	1.16±0.01	0.77±0.02	100.0±0.0	100.0±0.0	**81.8±0.5**	**92.2±0.2**	**52.7±1.0**	**81.2±0.4**	**40.3±0.9**	**68.7±0.6**
$-\mathcal{L}_{\text{Con}}$	0.65±0.01	1.16±0.01	0.77±0.02	100.0±0.0	100.0±0.0	77.2±0.5	89.9±0.3	51.1±1.0	80.3±0.4	33.9±2.0	65.2±1.3
−KG	0.72±0.01	1.68±0.01	1.12 ± 0.01	99.9±0.1	99.9±0.1	71.0±6.3	87.1±3.5	48.3±7.2	78.4±4.0	35.6±5.6	65.7±3.9
−KG & $\mathcal{L}_{\text{Con}}$	0.72±0.01	1.68±0.01	1.12 ± 0.01	93.7±1.6	97.5±0.8	73.2±0.5	87.8±0.4	49.5±0.2	79.9±0.2	37.1±1.2	66.7±0.0

C.3. Details of Reasoning-policy Structure

Table A.13 shows detailed experiment results of Table 4. DEDER ensures real-time inference speeds across various off-the-shelf devices while maintaining consistently superior performance compared to other ablation comparisons.

Table A.13. Details of reasoning-policy structure

Method	Inference Time			Train		Seen		Unseen Spatial		Unseen Environment	
	RTX 3090	RTX 3050	RTX 2080 Ti	SR	GC	SR	GC	SR	GC	SR	GC
Iterative	2.92±0.02	3.08±0.01	3.2±0.01	100.0±0.0	100.0±0.0	76.0±0.7	89.8±0.2	**54.0±0.4**	**81.4±0.2**	35.1±2.0	65.9±1.7
DEDER	0.65±0.01	1.16±0.01	0.77±0.02	100.0±0.0	100.0±0.0	**81.8±0.5**	**92.2±0.2**	52.7±1.0	81.2±0.4	**40.3±0.9**	**68.7±0.6**
$-\Psi_c$	0.63±0.01	1.47±0.01	0.74±0.01	99.9±0.1	99.9±0.0	75.9±0.3	89.6±0.1	52.2±0.2	80.7±0.1	29.9±1.0	63.8±0.5
$-\Psi_g$	0.63±0.01	1.16±0.01	0.74±0.01	100.0±0.0	100.0±0.0	76.8±0.3	89.6±0.1	50.9±0.1	79.3±0.1	30.5±2.6	62.9±1.1
$-\Psi_c\&\Phi_g$	0.60±0.01	1.00±0.01	0.64±0.09	100.0±0.0	100.0±0.0	63.3±0.1	83.1±0.1	39.5±0.4	73.5±0.4	28.7±1.0	61.2±0.4
DEDER w Ψ_a	0.63±0.01	1.16±0.01	0.75±0.01	99.9±0.1	99.9±0.0	74.7±0.4	88.6±0.2	52.2±0.2	80.7±0.1	35.1±1.0	64.1±0.8

C.4. Details of DEDER performance w.r.t. Policy Network Sizes

Table A.14 shows detailed experiment results of Table 5. As the network size increases, overall performance generally increases. However, the network capacity of the reasoning-policy Φ_R has a more significant impact on enhancing performance compared to the planning-policy Φ_P.

Table A.14. Details of DEDER performance w.r.t. policy network sizes

Φ_R	Φ_P	Parameter Size	Train		Seen		Unseen Spatial		Unseen Environment	
			SR	GC	SR	GC	SR	GC	SR	GC
t5-small	gpt2	0.06B+0.1B	100.0±0.0	100.0±0.0	81.8±0.5	92.2±0.2	52.7±1.0	81.2±0.4	40.3±0.9	68.7±0.6
t5-small	gpt2-medium	0.06B+0.4B	100.0±0.0	100.0±0.0	81.4±0.1	92.1±0.1	52.6±0.4	81.0±0.4	39.7±1.2	69.9±0.0
t5-small	gpt2-large	0.06B+0.8B	100.0±0.0	100.0±0.0	81.8±0.1	92.1±0.1	52.5±0.2	81.2±0.1	67.5±1.7	39.7±2.0
t5-base	gpt2	0.2B+0.1B	100.0±0.0	100.0±0.0	79.4±0.5	91.2±0.3	55.2±0.7	82.8±0.4	41.8±1.7	69.3±1.2
t5-base	gpt2-medium	0.2B+0.4B	100.0±0.0	100.0±0.0	78.8±0.6	91.0±0.3	55.2±0.3	83.1±0.3	42.0±1.0	69.6±0.3
t5-base	gpt2-large	0.2B+0.8B	100.0±0.0	100.0±0.0	80.0±1.3	91.5±0.5	54.9±0.5	82.4±0.6	43.1±2.4	70.2±1.3
t5-large	gpt2	0.7B+0.1B	100.0±0.0	100.0±0.0	**82.1±1.0**	**92.3±0.3**	57.2±0.4	83.0±0.1	47.7±1.0	73.5±0.8
t5-large	gpt2-medium	0.7B+0.4B	100.0±0.0	100.0±0.0	81.8±0.6	**92.3±0.2**	**57.5±0.4**	**83.2±0.2**	48.3±1.4	73.5±0.8
t5-large	gpt2-large	0.7B+0.8B	100.0±0.0	100.0±0.0	81.2±0.4	92.0±0.1	**57.5±0.3**	83.1±0.1	**49.1±1.2**	**74.2±0.3**

D. Embodied Knowledge Graph Examples

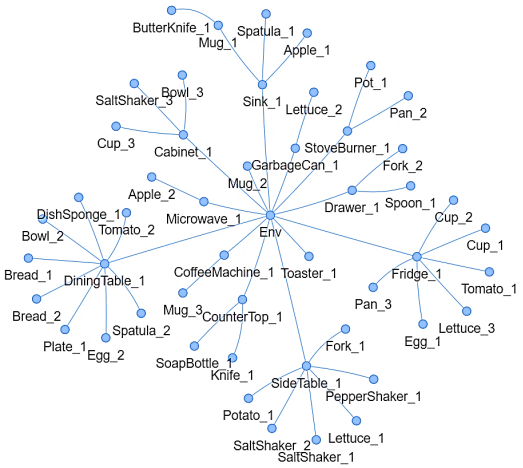


Figure A.6. Example of initial embodied kG

Figures A.6 and A.7 show examples of our embodied KG. Initially, the task description provides information on a partially observable environment. From this information, our sLM-based policy efficiently encapsulates knowledge of the environment

via an embodied KG as illustrated in Figure A.6. As the agent interacts with its surroundings, gathering information necessary for task completion, the embodied KG reflects those changes via the update function U, as specified in (5). Examples of these dynamic changes in the embodied KG, reflective of the evolving environment, are depicted in FigureA.7. Finally, our KG retriever function V, detailed in (6), selects a subset of the embodied KG based on the task description h and observation o. This subset is then used for embodied KG prompting to our reasoning-policy, as illustrated in Figure A.8.

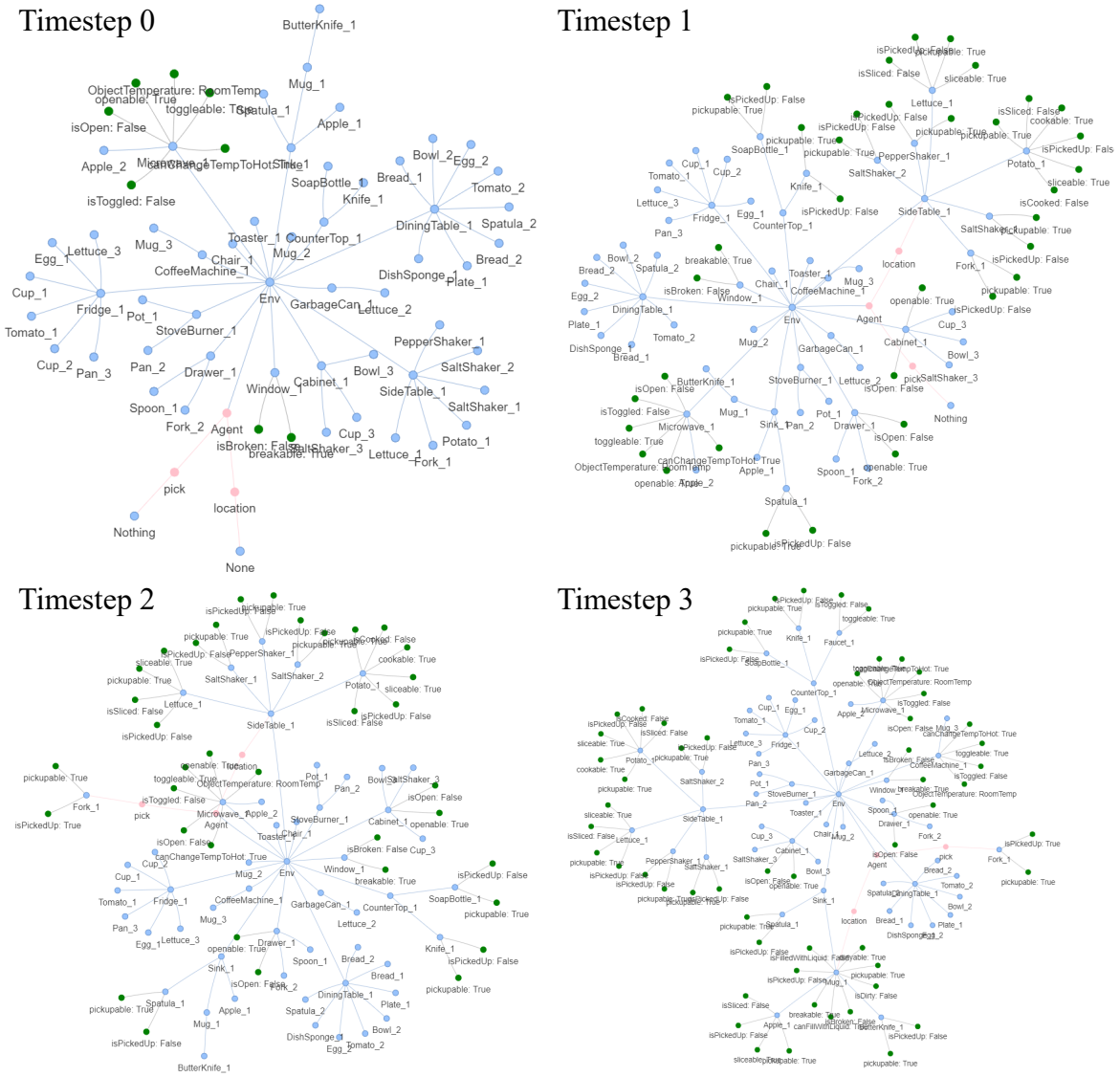


Figure A.7. Maintenance of the embodied KG to accommodate changes in environment information during task execution by the agent.

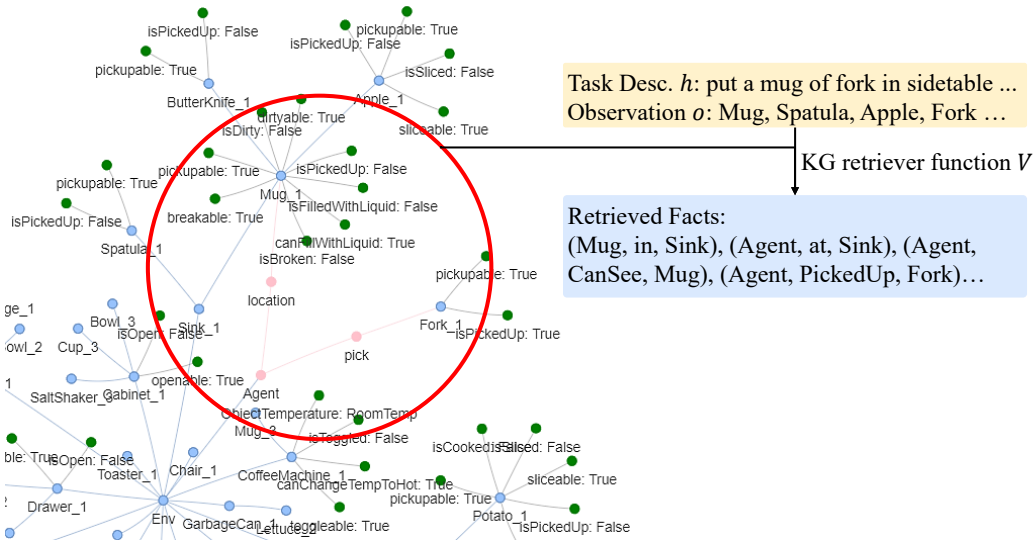


Figure A.8. Example of retrieved subset of embodied KG