LLM-Planner: Few-Shot Grounded Planning for Embodied Agents with Large Language Models

Chan Hee Song
The Ohio State University
song.1855@osu.edu

Jiaman Wu
The Ohio State University
wu.5686@osu.edu

Clayton Washington
The Ohio State University
washington.534@osu.edu

Brian M. Sadler
DEVCOM ARL
brian.m.sadler6.civ@army.mil

Wei-Lun Chao
The Ohio State University
chao.209@osu.edu

Yu Su
The Ohio State University
su.806@osu.edu

Abstract

This study focuses on embodied agents that can follow natural language instructions to complete complex tasks in a visually-perceived environment. Existing methods rely on a large amount of (instruction, gold trajectory) pairs to learn a good policy. The high data cost and poor sample efficiency prevents the development of versatile agents that are capable of many tasks and can learn new tasks quickly. In this work, we propose a novel method, LLM-Planner, that harnesses the power of large language models (LLMs) such as GPT-3 to do few-shot planning for embodied agents. We further propose a simple but effective way to enhance LLMs with physical grounding to generate plans that are grounded in the current environment. Experiments on the ALFRED dataset show that our method can achieve very competitive few-shot performance, even outperforming several recent baselines that are trained using the full training data despite using less than 0.5% of the paired training data. Existing methods can barely complete any task successfully under the same few-shot setting.

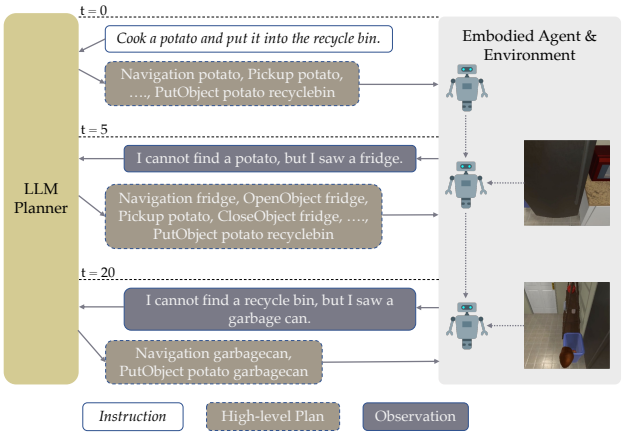

Figure 1. An illustration of LLM-Planner for high-level planning. After receiving the natural language instruction ($t = 0$), LLM-Planner first generates a high-level plan by prompting a large language model (e.g., GPT-3). When the embodied agent gets stuck during the execution of the plan, we re-plan based on observations from the current environment to generate a more grounded plan, which may help the agent get unstuck.

1. Introduction

We study embodied agents that follow natural language instructions to carry out complex tasks in a partially observable, visually-perceived environment. Recent years have seen surging interests in such embodied instruction following agents [9] based on simulated environments such as Matterport3D [2] and AI2-Thor [15]. Contemporary agents rely on pairs of natural language instructions and demonstrations for learning, which map the instruction (e.g., *"cook a potato"*) to the desired trajectory (i.e., the sequence of primitive actions such as [RotateRight, MoveAhead, Pickup potato, ...]) in a certain environment and a starting state. However, such paired data is highly costly to collect, while contemporary agents require a large amount of paired data for learning a task (e.g., thousands of paired examples from the ALFRED dataset [32] for each type of task). In contrast, humans can learn new tasks with very few demonstrations. In this paper, we investigate the question: *How to dramatically improve the sample efficiency of embodied agents so that they can quickly learn a task with only a few paired examples?*

Towards this goal, our first key realization is that natural language instructions do not need to be directly paired with low-level trajectories if we adopt hierarchical planning models (e.g., [31, 35]), which consist of a *high-level planner* and a *low-level planner*. The high-level planner

maps the instruction into a *high-level plan* (HLP), which is a sequence of abstract subgoals (e.g., [Navigation potato, Pickup potato, Navigation microwave, ...]) that the agent needs to reach, in the specified order, to accomplish the final goal. The low-level planner then maps each subgoal into a sequence of primitive actions for accomplishing that subgoal in the current environment and state. *Given a high-level plan, low-level planning is conditionally independent of the natural language instruction*. It becomes the traditional object localization and navigation problem [6] (for navigation subgoals) or simply executing the specified interaction action with the right objects (for interaction subgoals). The low-level planner can be trained with data synthesized from the simulator (see, e.g., [25]). Only the high-level planner needs paired training data in the form of (instruction, HLP).

However, the high-level planner could still require a significant amount of paired training data, especially if we aim for versatile agents that are capable of many different tasks. Intuitively, high-level planning is closely related to commonsense reasoning [36]. In the previous example, the high-level planner's task is to infer that *"cooking a potato"* entails first *"go find a potato,"* then *"pick up the potato"* and *"go find an object, e.g., a microwave, that can be used for cooking,"* and so on. Our second key realization is that, since large language models (LLMs) such as GPT-3 [4] are shown to have captured tremendous commonsense knowledge [36], they can potentially be used as *few-shot high-level planner* for embodied agents.

To this end, we propose LLM-Planner, the first LLM-based high-level planner for embodied instruction following. Specifically, we follow the in-context learning paradigm [4, 20] and only use a small number of paired examples. In addition, no parameter update is needed, which saves development time.[1] For the example in Figure 1, at the beginning of an episode ($t = 0$), given a natural language instruction, we directly prompt an LLM to generate the HLP by giving it several exemplar pairs of (instruction, HLP) in its context. We also leverage established techniques such as dynamic in-context example retrieval [8, 19, 27, 29] and logit biases [10] to further improve the in-context learning performance.

While the HLPs generated by LLMs are already reasonable, they still lack a fundamental aspect of embodied agents — *physical grounding*; i.e., the generated HLP needs to be grounded to the environment the agent is in. Figure 1 demonstrates two common types of planning errors due to lack of grounding: 1) *Object mismatch* ($t = 20$): an object is denoted as garbage can in the environment but the human user refers to it as *"recycle bin."* 2) *Unattainable plans* ($t = 5$): the HLP requires finding a potato, but the only potato in the environment is stored in a closed fridge; the agent will wander endlessly as a result. We propose a novel *dynamic grounded re-planning* algorithm to empower LLM-Planner with physical grounding. Specifically, as an agent is executing the initial HLP, whenever it has taken too many steps to reach the current subgoal or has made too many failed attempts, we dynamically prompt the LLM again to generate a new continuation of the partial HLP that has been completed at that point. For grounding, we add the list of objects perceived in the environment so far into the prompt as a simple but effective description of the current environment. For the example in Figure 1, at $t = 5$, the agent is taking too long to find a potato. It re-prompts the LLM with the object fridge observed in the environment, and LLM-Planner generates a new HLP from scratch (because no subgoal has been completed so far) that directs the agent to look for a potato in the fridge. This is feasible because LLMs may have the commonsense knowledge that food like potatoes are often stored in a fridge, but we need to inform the LLM that there exists a fridge in the current environment. As another example, at $t = 20$, after completing all the subgoals in the HLP but the last one, the agent cannot find the required recycle bin but has observed a garbage can. LLM-Planner thus generates a new continuation of the already-completed partial HLP, which consists of the only remaining subgoal.

We evaluate LLM-Planner on the standard ALFRED [32] household dataset by integrating it with the perception and low-level planner from a strong baseline model, HLSM [3]. Using less than 0.5% paired training data, we show that LLM-Planner achieves comparable performance to HLSM and outperforms multiple other baselines, which are trained with the full training set. Under the same few-shot setting, existing methods can barely complete any task successfully. Our work opens a new door for developing extremely sample-efficient embodied agents by harnessing the power of large language models and grounding.

2. Preliminaries

Vision-and-Language Navigation. Embodied instruction following is often also referred as vision-and-language navigation (VLN), though it additionally involves interaction actions and usually features a much longer time horizon than typical VLN tasks (e.g., Room2Room [2]). To be consistent with the literature, we will use these two terms interchangeably. We will primarily focus on the standard ALFRED [32] dataset, which is built on top of the AI2-Thor [15] simulator, but our method can easily generalize to other datasets and environment. We choose ALFRED mainly considering its diversity in task types (7 different task types) and long-horizon tasks (on average 50 actions per task).

The VLN task is defined as following: Given a language

[1] In the experiments we use 100 (instruction, HLP) pairs compared with 21,023 (instruction, trajectory) pairs used by existing methods.

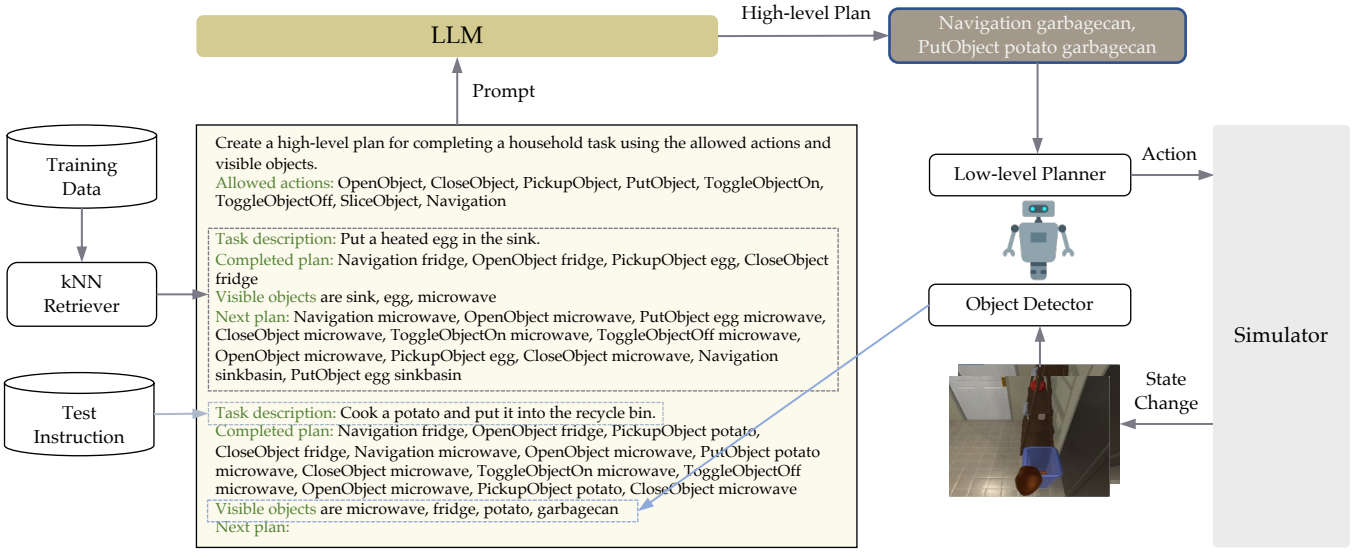

Figure 2. Overview of LLM-Planner with prompt design and dynamic grounded re-planning.

instruction I, an agent needs to predict and carry out a sequence of primitive actions in the environment E to accomplish the task. In datasets like ALFRED [32], the instruction I consists of a high-level goal I_H and (optionally) a list of step-by-step instructions I_L, as exemplified in Figure 2. A VLN task can thus be represented by a tuple (I, E, G), where G is the goal test of the task. We consider hierarchical planning models [35] for VLN, which is explored to various extent in several recent studies [3, 25, 31, 33], but none of them considers the few-shot setting or LLMs for planning. In this formulation, planning is modeled in a hierarchical fashion. The high-level planner maps the instruction I into a high-level plan (HLP) $L_h = [g_0, g_1, \cdots, g_T]$, where each subgoal g_i is specified as (high-level action, object). We define a high-level action to be a collection of primitive actions that can complete a single goal-condition in ALFRED [32]. However, while ALFRED contains primitive locomotive actions, e.g., (MoveForward, 25) and (RotateRight, 90), with specifiable changes in magnitude, all the interaction actions are abstracted into a single action, e.g., (PickupObject, knife), without any changes to the agent's posture. Therefore, the high-level action space consists of 1 navigation action (Navigation) and 7 interaction actions (PickupObject, PutObject, OpenObject, CloseObject, ToggleOnObject, ToggleOffObject, SliceObject).

The low-level planner maps each subgoal g_i into a sequence of primitive actions $L_l^i = \left[a_i^0, a_i^1, \cdots, a_i^{T_i}\right]$. State-of-the-art VLN methods [3, 25] use a map-based low-level planner and a simple path-finding algorithm to find the target object in the current subgoal from the map. It is important to note that, once the high-level plan L_h is specified, the low-level planning becomes independent of the instruction I. More formally, $P(L_l|I, L_h, E) = P(L_l|L_h, E)$. All the components involved in the low-level planner are either deterministic or trained using synthetic data from the simulator. No paired data involving instructions is needed.

In-Context Learning/Prompting. Recently, in-context learning (also known as prompting) [4] have drawn great attention with the rise of LLMs. By designing different prompts, LLMs can be adapted to different downstream tasks with a few examples as demonstration without updating any of the parameters. In this work, we explore in-context learning with LLMs for embodied agent planning.

True Few-Shot Setting. While only using a small number of training examples, many few-shot studies use a large validation set for prompt design and model selection [4]. Recent studies [27] have shown that such large validation sets are responsible for overestimation of the efficacy of language models because they create a strong bias for model selection and violate the intended few-shot setting. To avoid such bias, we adhere to the true few-shot setting [27] in which prompt design and model selection is conducted via cross-validation on the same small training set instead of using a separate validation set.

3. LLM-Planner

In this section, we describe our method LLM-Planner, which leverages LLMs such as GPT-3 to do few-shot grounded high-level planning for embodied agents.

3.1. Overview

LLMs such as GPT-3 are pre-trained to generate natural language. To adapt them as high-level planners, the first

step is to design an appropriate prompt to guide them to generate high-level plans. We discuss our prompt design in Section 3.2. The choice of in-context examples is critical for the performance of LLMs, and recent works [8, 27] have shown that dynamically retrieving similar examples for each test example is beneficial. We adopt a k-nearest-neighbor (kNN) retriever to select the in-context examples (Section 3.3). We also use logit biases [10] to further constrain the output space of the LLM to the allowed set of actions and objects (Section 3.4). With all the above designs, we have obtained the *static* version of LLM-Planner, which can already generate reasonable HLPs. In Section 3.5, we propose a novel dynamic grounded re-planning algorithm to enhance LLMs with the ability to ground to the current environment, which further improves the HLP quality. Finally, we discuss how to integrate LLM-Planner into existing embodied agents to empower them with few-shot planning capabilities in Section 3.6. An overview of LLM-Planner is shown in Figure 2.

3.2. Prompt Design

While GPT-3 is shown to be a powerful few-shot learner in a variety of tasks, its power can only be unleashed with carefully designed prompts that are tailored for the desired behavior. The final HLP quality can be sensitive to minor design choices in the prompt (e.g., how the HLP is presented, or sometimes even the choice of punctuation). Therefore, we identify core components of the prompt and systemically compare different design choices under the true few-shot setting based on leave-one-out cross-validation (LOOCV). The evaluations for some of the key design choices are discussed in Section 4.5 and 4.6.

Our final optimal prompt is shown in Figure 2. The prompt begins with an intuitive explanation of the task and the list of allowable high-level actions. It is then followed by the in-context examples selected by the kNN retriever (Section 3.3). When we provide only the high-level goal instruction to GPT-3, we use the format "Task description: [high-level goal instruction]." When we include the step-by-step instructions, we include another line "Step-by-step instructions: [step-by-step instructions]" following the goal instruction. For dynamic grounded re-planning (Section 3.5), we add the subgoals that have been completed and the list of objects observed so far in the environment after the task description. Finally, we append the test example in the same format that ends with "Next plan:".

3.3. In-context Example Retrieval

The in-context examples are an important source of task-specific information for the LLM. Different examples could provide different information for the current task. Intuitively, if the current task is to *"cook a potato,"* an in-context example that demonstrates the HLP for *"cooking an egg"* is likely more informative than one that demonstrates how to *"clean a plate."* Specifically, we use a frozen BERT-base model [7] to evaluate the pairwise similarity between each training example and the current test example. The similarity of two examples is defined based on the Euclidean distance between the BERT embedding of their corresponding instruction. For each test example, we then retrieve the K most similar examples from the small set of paired training examples we have, where K is a hyperparameter that we tune under the true few-shot setting (Section 4.6).

Algorithm 1 Dynamic Grounded Re-planning with LLM-Planner

$I \leftarrow$ Instruction
$O \leftarrow$ Set of observed object
$G \leftarrow$ List of completed subgoals so far
$S \leftarrow$ LLM-Planner(I, O, G) ▷ Full HLP
$t \leftarrow 0$ ▷ Time step
$k \leftarrow 0$ ▷ Subgoal index
$s \leftarrow S[k]$ ▷ First subgoal
$a_t \leftarrow$ Low-Level-Planner(s) ▷ First action
while k <len(S) **do**
  execute a_t
  $O_t \leftarrow$ Object-Detector(current camera input)
  O.insert(O_t)
  if current subgoal s fails or after n time steps **then**
    $S \leftarrow$ LLM-Planner(I, O, G) ▷ New HLP
    $k \leftarrow 0$
    $s \leftarrow S[k]$
  else if current subgoal s is completed **then**
    $k \leftarrow k + 1$
    $s \leftarrow S[k]$ ▷ Get next subgoal
  end if
  $t \leftarrow t + 1$
  $a_t \leftarrow$ Low-Level-Planner(s)
end while

3.4. Logit Biases

Since the HLPs that LLM-Planner needs to predict consist of tokens from a fixed set of actions and objects, we leverage the *logit bias* option of the GPT-3 API to constrain the output space. By providing a list of allowable actions and objects to GPT-3, we can enforce it to allocate the majority of its probability mass during generation to those tokens and reduce the the generation of invalid tokens. The strength of this constraint is controlled by the value of the logit bias, with a larger value indicating a stronger constraint. Specifically, we add the same logit bias to all tokens in the list of actions, visible objects (for dynamic planning), or all possible objects (for static planning), and the value is selected through cross-validation.

3.5. Dynamic Grounded Re-planning

Using LLM-Planner as a *static* high-level planner that only predicts an HLP at the begining of a task already shows good data efficiency and accuracy. As discussed earlier, however, such static planning lacks physical grounding to the environment and can lead to incorrect objects and unattainable plans (Figure 1). When such issues happen, the agent cannot complete the current subgoal specified in the HLP, which will lead to one of two possible situations: 1) it fails to execute an action (e.g., bumping into a wall or failing to interact with an object), or 2) it takes a long time and still has not completed the current subgoal (e.g., wandereing endlessly). Intuitively, knowing the objects in the current environment can be very helpful for addressing both of these issues. For example, knowing that there is a fridge, the LLM may produce an HLP that directs the agent to go to the fridge and try to find a potato in that, because it may have learned the commonsense knowledge that food is likely stored in a fridge.

To this end, we present a simple but novel way to enhance LLM-Planner with physical grounding by injecting a list of observed objects, which may be detected using the object detector of the embodied agent, from the environment into the prompt (Figure 2). We add logit biases to these observed objects so LLM-Planner can prioritize producing a plan with those objects if they are relevant for the current task.

With such flexible grounded re-planning capability, we further propose a dynamic grounded re-planning algorithm (Algorithm 1) to dynamically update the HLP during the course of completing a task. This is in contrast with most existing work that adopts a similar hierarchical planning model [25], which only predicts a fixed HLP up front and sticks to that no matter what happens during the execution. In our algorithm, re-planning will be triggered under either of two conditions: 1) the agent fails to execute an action, or 2) after a fixed number of time steps. A new continuation of the already-completed partial HLP will be generated LLM-Planner based on the observed objects, and the agent will carry on with the new plan, which may help it get unstuck.

3.6. Integration with Existing VLN models

We now discuss how to integrate LLM-Planner with the existing models to empower them with the few-shot planning capability. LLM-Planner provides a fairly generic and flexible interface for integration. As shown in Algorithm 1, it only needs the embodied agent to provide an object list and has a low-level planner that can turn the predicted HLP into low-level actions. It has no assumption about the inner working of the agent. For evaluating the end-to-end task completion performance of LLM-Planner, we integrate it with a strong baseline method, HLSM [3], which satisfies such an interface.

4. Experiments

4.1. Dataset

We evaluate the efficacy of LLM-Planner in generating high-level plans using the ALFRED [32] benchmark, a vision-and-language navigation dataset that requires embodied agents to follow instructions and use visual input to complete tasks in a simulated, spatially continuous household environment. The dataset consists of 7 task types, ranging in difficulty from *moving a single object to a new location* to *placing a heated slice of an object into a receptacle*. Each task is accompanied by human-written annotations of a high-level goal and a series of more granular step-by-step instructions, created by human annotators as they watched expert demonstrations of the tasks. Due to the noise in the natural language instructions and the complexity of planning required to complete such long-horizon tasks, ALFRED is a challenging test of an embodied agent's ability to produce robust and accurate plans. We use stratified random sampling to choose 100 training examples that are representative of the 7 task types.

4.2. Metrics

We report two main metrics used by ALFRED and one metric created by us to calculate the high-level planning accuracy. Success rate (SR) is the percentage of tasks fully completed by the agent. A task is only considered complete when all the subgoals are completed. Therefore, each task can either be successfully or unsuccessful and success rate does not take account of a partially succeeded task. Goal-condition success rate (GC) is the percentage of completed goal-conditions. Goal-conditions are defined as collection of state changes necessary to complete the task. For example, in the task *"Slice a heated bread"*, bread being sliced and bread being heated are both goal-conditions.

To directly evaluate high-level planning, we introduce a new metric named high-level planning accuracy (HLP ACC), i.e., the accuracy of the predicted HLP compared to the ground-truth HLP. For the static planning setting, we compare the generated HLP with the ground-truth HLP and deems a plan as incorrect if it does not perfectly match the ground truth, and correct otherwise. For the dynamic planning setting, we report a range since we do not know what the future HLP will be if the low-level planner/controller fails to execute a subgoal. Specifically, the upper bound of the range is obtained by treating the an HLP as correct if it perfectly matches a certain prefix of the ground truth. The lower bound is obtained in the same way as the static planning setting.

4.3. Implementation Details

We choose 100 examples for our LLM-Planner among 21,023 ALFRED training examples. We apply stratified

Model	Test Unseen		Test Seen		Valid Unseen			Valid Seen		
	SR	GC	SR	GC	SR	GC	HLP ACC	SR	GC	HLP ACC
Full-data setting: 21,023 (instruction, trajectory) pairs										
Goal instruction only										
HiTUT [38]	11.12	17.89	13.63	21.11	10.23	20.71	–	18.41	25.27	–
HLSM [3]	20.27	27.24	25.11	35.79	**18.28**	**31.24**	31.24 – 70.17	29.63	38.74	38.74 – 77.64
Step-by-step instructions										
E.T. [26]	8.57	18.56	**38.42**	**45.44**	7.32	20.87	–	**46.59**	**52.92**	–
HiTUT [38]	13.87	20.31	21.27	29.97	12.44	23.71	–	25.24	34.85	–
M-TRACK [33]	16.29	22.60	24.79	33.35	17.29	28.98	–	26.70	33.21	–
FILM [25]	27.80	**38.52**	28.83	39.55	–	–	54.93	–	–	60.86
LEBP [18]	**28.30**	36.79	28.97	36.33	–	–	–	–	–	–
Few-shot setting: 100 (instruction, high-level plan) pairs										
Goal instruction only										
LLM-Planner (Static) + HLSM	11.58	18.47	13.05	20.58	11.10	22.44	28.67	11.82	23.54	27.45
LLM-Planner + HLSM	13.41	22.89	15.33	24.57	12.92	25.35	33.81 – 55.85	13.53	28.28	35.08 – 54.33
Step-by-step instructions										
HLSM [3]	0.61	3.72	0.82	6.88	0.00	1.86	0.00	0.13	2.82	0.00
FILM [25]	0.20	6.71	0.00	4.23	0.00	9.65	0.00	0.00	13.19	0.00
LLM-Planner (Static) + HLSM	15.83	20.99	17.87	23.10	14.26	26.12	43.24	15.84	25.43	39.87
LLM-Planner + HLSM	**16.42**	**23.37**	**18.20**	**26.77**	**15.36**	**29.88**	46.59 – 68.31	**16.45**	**30.11**	50.33 – 71.84

Table 1. Main results on the ALFRED dataset. "(Static)" means the static planning setting, otherwise it is the default dynamic grounded planning setting. Some methods support using only the goal instruction or additionally using the step-by-step instructions. We compare under both configurations. LLM-Planner achieves a strong performance that is competitive with existing methods despite using less than 0.5% of paired training data. Under the same few-shot setting, existing methods can barely complete any task successfully.

random sampling to ensure we have a fair representation of all 7 task types in the 100-example set. For the kNN retriever, we use the pretrained BERT-base-uncased model from the Huggingface Transformers Library [37]. For the LLM, we use the public GPT-3 [4] API with 9 in-context examples chosen from the 100 training examples by the kNN retriever. We set the temperature to 0 and apply a logit bias of 0.1 to all allowable output tokens as described in Section 3.4. The object list for dynamic grounded re-planning is retrieved from the object detector. Specifically, we use the pretrained object detector from HLSM's perception model to retrieve the label of all detected objects. We limit the number of detected objects to 10 and only include objects with a label confidence more than 80% to reduce noise. It is worth noting that we can potentially use any object detector to obtain the object list, and we only use HLSM's perception model to save computation cost and time. To avoid violating our few-shot assumption, we use the pretrained navigation, perception, and depth model from HLSM which are trained using only synthesized trajectories from the simulator, without any paired training data involving natural language instructions or human annotations.

	LOOCV HLP accuracy
Best Model	40.59
– kNN Retriever	17.48
– Logit Biases	38.10
– Both	13.43

Table 2. LLM-Planner's component ablation using LOOCV.

4.4. Main Results

The main results are shown in Table 1. We first compare the performance of HLSM when using our LLM-Planner as the high-level planner compared with its native version, which is trained using the full training set of ALFRED. We find that LLM-Planner's few-shot performance is competitive to the original HLSM, and outperforms several recent baselines such as E.T., HiTUT, and M-TRACK, despite using less than 0.5% of paired training data. On the other hand, when trained using the same 100 examples (i.e., re-training HLSM's high-level planner), HLSM (and FILM as well) can barely complete any task successfully. We leave the other modules, such as navigation, depth estimation, and perception, intact since they do not use the paired data involving natural language instructions. Furthermore, we

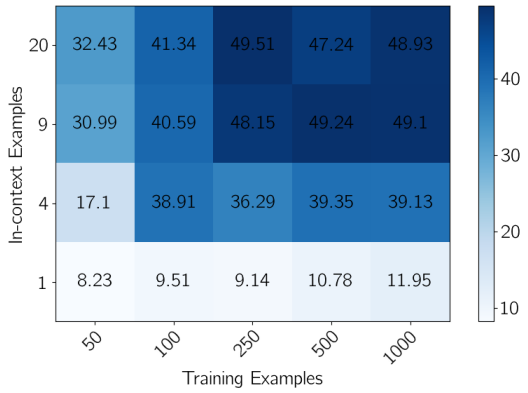

Figure 3. LOOCV HLP accuracy results for number of in-context examples vs. number of training examples

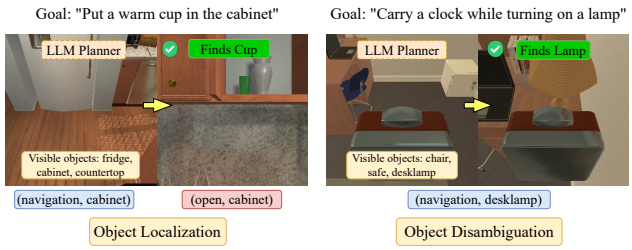

Figure 4. Case studies for LLM-Planner

see a considerable improvement from dynamic grounded re-planning over static planning, especially in the goal instruction only setting, where it improves 1.83% SR in the unseen test split. This confirms the effectiveness of the dynamic grounded planning. But we also note that there is still a large room for further improvement.

4.5. Ablation Study for LLM-Planner

General Ablation We ablate on different components of LLM-Planner to demonstrate the importance of techniques to use to improve the LLM-Planner's performance. Mainly we ablate on the using the kNN retrieval and the logit biases for GPT-3. We follow the LOOCV process and use only the high-level planning accuracy to determine our choices. Results from this study are in Table 2. We first ablate the kNN retriever module, by replacing it with a retriever that randomly selects in-context examples from the 100 example set. As we can see from Table 2, removing kNN retriever and letting it select random (and often unrelated) examples lead to a significant decrease in HLP accuracy. This is expected because the getting relevant in-context examples is crucial to GPT-3 because it will bias it's HLP prediction to follow similar trajectory as the relevant in-context examples.

Furthermore, we found out that enabling logit biases to favor objects that appear in the environment lead to a decent boost in the high-level planning accuracy. By having GPT-3 favor objects that appear in the environment makes GPT-3 more robust in the cases where the instruction is ambiguous or objects are referred as different names. For example, for an instruction *"Turn on the lamp"*, GPT-3 have no idea what type of lamp does the instruction means. By enabling the logit biases to favor objects that appear in the environment (e.g. TableLamp), we can correctly guide GPT-3 to output (TurnOnObject, TableLamp). Another example is when the instruction refers to RecycleBin but the object name used in the environment is GarbageCan. In this case, having objects name from the environment as logit biases can correctly guide the GPT-3 to output the relevant and correct objects.

For each setting including plans generated by LLM-Planner, temperature, and K (number of in-context examples) hyperparameters were tuned using 5-fold cross-validation on the training set of 100 examples chosen by stratified random sampling to ensure all task types are represented in the set.

KNN Retriever Ablation The embedding type for the kNN retrieval module is a BERT sentence embedding of the high-level instructions in settings that omit step-by-step instructions and a BERT sentence embedding of concatenated high-level and step-by-step instructions in the settings that include step-by-step instructions. BERT embeddings were selected over RoBERTa [21] embeddings through LOOCV on the small training set in which the BERT embedding module retrieved at least one in-context example with the same ground-truth action sequence as the held-out example at a higher rate than the RoBERTa embedding module.

4.6. Fine-grained Analyses

Effect of Number of Examples To validate our choice of number of training examples for the kNN retriever and number of in-context examples for the prompt, we perfrom LOOCV on different number of both hyperparameters. As shown in Figure 3, we find out that the HLP accuracy reaches a ceiling around 500 examples and that simply increasing the training example size does not lead to the improvement in HLP accuracy. Furthermore, we find that the optimal number of in-context examples is 9. Although adding more in-context examples do not usually hurt the performance, we found out that it is not meaningful enough to justify additional computational cost.

Case Studies We show 2 examples where LLM-Planner does object localization and disambiguation in the environment in Figure 4. For the first case, even using only the high-level goal instruction, the LLM-Planner could successfully predict that the cup is likely to be located in the cabinet after failing to pick up the cup in the environment.

This shows the semantic knowledge of the LLM-Planner and comparable to what semantic map tries to achieve in FILM [25]. For the second case, we show that the LLM-Planner can correctly ground the word *"lamp"* to the desklamp in the environment by using the grounding environment (visible object list) as a part of the prompt to the LLM-Planner.

5. Related Work

5.1. Vision-and-language Navigation

As vision-and-language navigation task gets more complex, various models have been proposed that shifted the trends from the existing single-neural network model architectures. In navigation-only datasets such as R2R [2], models that generate the output action sequence end-to-end by utilizing a Transformer achieves a high performance [26, 34]. Recently, with the emergence of a BERT and its derivatives [4, 7, 21], the instructions are encoded by BERT-based models with a higher generalizability for downstream navigation task [11, 17, 23, 24]. These models jointly learns the linguistic and visual representations with cross-attention to be grounded on the environment.

However, in the navigation and interaction dataset such as ALFRED [32], models [3, 18, 25] that separated the high-level and low level planning interface (hierarchical planning models) have been dominating the leaderboard. These models use a pretrained BERT to generate high-level plans and leverage a semantic map to guide the agent achieving the target objects in the high-level plans for low-level planning.

This modular approach has shown strong performance where the training data has been hard to collect since each modular component can be trained independently to further improve the performance. $(SL)^3$ [31] only uses 10% of annotated data to learn how to generate natural language subtasks with goals and then match admissible actions to subtasks. We take this modular approach one step further and propose to use a few-shot setting on large language model (LLM) to conduct high level planning and utilize it's rich pre-trained information.

5.2. General In-context Learning and Prompting

In-context learning [4] uses a small number of examples in prompts as a sample for the task to LLM without requiring additional fine-tuning for different tasks. The way to choose the optimal in-context examples is also important aspect of in-context learning [19]. However, it is unclear how to expand the single-modal LLM in-context learning to multi-modality. In this paper, we present a novel way of grounding environment to the LLM in a form of object list.

Studies have shown that the LLM is very sensitive to the prompt design, especially in few-shot or zero-shot setting [14, 27]. Simply adding the significant keywords, such as a description of task, may increase the performance dramatically [14]. We follow the true few-shot setting proposed in [27] which does not use an additional validation set to choose prompt templates or hyperparameters. This setting shows the strength of our approach without requiring any additional data at all.

5.3. Prompting for VLN

The use of large language model for decision making has been a increasingly popular topic due to the amazing capability of the large language models to perform multi-task generalization with a limited number of training data. Previous works have either pre-trained a large Transformer model on image-text data [30] or applied an additional re-ranking or alignment to the generated plan [22, 31].

JARVIS [39] applies few-shot learning by fine-tuning BART [16] with free-form dialogues and completed high-level plans to generate the next high-level plan. PROG-PROMPT [12] utilizes the LLM (e.g., GPT-3 [4] and Codex [5]) to format the instructions as program-like prompt and generate an executable plan. LM-Nav [30] prompts LLM with raw navigation instructions and 3 examples to generate a list of landmarks for a vision-language model to infer a joint probability distribution over landmarks and images. Language planner [13] asks sufficient large language model (GPT-3) to generate an free-formed instructions given a prompt with an in-context example and a goal. A pretrained BERT-derivative converts the free-formed instructions instructions to an admissible action, which is later appended to the prompt for the future plan. However, LLM-Planner can do this transformation in a single-step and does not require an intermediate model that transforms the free-formed instructions into admissible actions. Saycan [1] ranks the score of a list of pre-defined admissible action from LLM by prompting in-context examples, combining with an affordance function which assigns higher weights to the objects appearing in the current scene.

Existing works either uses LLM as a static generator for HLP [5, 30, 39] or to translate instructions into concise/admissible instructions [1, 13]. On the other hand, LLM-Planner grounds the environment to the LLM while dynamically planning long-horizon subgoals without needing to use any other intermediate models or list of admissible actions. With appropriate prompt design and in-context example retrieval, we show that LLM-Planner can generate complete high-level plans grounded on the environment without any post-filtering mechanisms to perform similarly compare to the given methods with only fraction of data.

6. Conclusion

We demonstrate that large language models (LLMs) can be used as a planner for embodied agents and can dramati-

cally reduce the amount of human annotations needed for learning the instruction following task. Our work opens a new door for developing extremely sample-efficient embodied agents by harnessing the power of large language models and enhancing them with physical grounding. Promising future directions include exploring other LLMs such as Codex [5], better prompt design, and more advanced methods for grounding and dynamic re-planning.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022. 8

[2] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sunderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, pages 3674–3683, 2018. 1, 2, 8

[3] Valts Blukis, Chris Paxton, Dieter Fox, Animesh Garg, and Yoav Artzi. A persistent spatial semantic representation for high-level natural language instruction execution. In *Conference on Robot Learning*, pages 706–717. PMLR, 2022. 2, 3, 5, 6, 8, 12

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. 2, 3, 6, 8

[5] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, David W. Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Babuschkin, S. Arun Balaji, Shantanu Jain, Andrew Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew M. Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374, 2021. 8, 9

[6] G.N. Desouza and A.C. Kak. Vision for mobile robot navigation: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):237–267, 2002. 2

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. 4, 8, 12

[8] Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online, Aug. 2021. Association for Computational Linguistics. 2, 4

[9] Jing Gu, Eliana Stefani, Qi Wu, Jesse Thomason, and Xin Wang. Vision-and-language navigation: A survey of tasks, methods, and future directions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7606–7623, 2022. 1

[10] Bernal Jiménez Gutiérrez, Nikolas McNeal, Clay Washington, You Chen, Lang Li, Huan Sun, and Yu Su. Thinking about gpt-3 in-context learning for biomedical ie? think again. *arXiv preprint arXiv:2203.08410*, 2022. 2, 4

[11] Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. A recurrent vision-and-language bert for navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1643–1653, June 2021. 8

[12] Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Visual language maps for robot navigation. *arXiv preprint arXiv:2210.05714*, 2022. 8

[13] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *arXiv preprint arXiv:2201.07207*, 2022. 8

[14] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022. 8

[15] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017. 1, 2

[16] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. 8

[17] Xiujun Li, Chunyuan Li, Qiaolin Xia, Yonatan Bisk, Asli Celikyilmaz, Jianfeng Gao, Noah A. Smith, and Yejin Choi. Robust navigation with language pretraining and stochastic sampling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1494–1499, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. 8

[18] Hao Liu, Yang Liu, Hong He, and Hang Yang. Lebp - language expectation & binding policy: A two-stream framework for embodied vision-and-language interaction task learning agents. *ArXiv*, abs/2203.04637, 2022. 6, 8

[19] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online, May 2022. Association for Computational Linguistics. 2, 8

[20] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*, 2021. 2

[21] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Ro{bert}a: A robustly optimized {bert} pretraining approach, 2020. 7, 8

[22] Lajanugen Logeswaran, Yao Fu, Moontae Lee, and Honglak Lee. Few-shot subgoal planning with language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5493–5506, Seattle, United States, July 2022. Association for Computational Linguistics. 8

[23] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in neural information processing systems*, 32, 2019. 8

[24] Arjun Majumdar, Ayush Shrivastava, Stefan Lee, Peter Anderson, Devi Parikh, and Dhruv Batra. Improving vision-and-language navigation with image-text pairs from the web. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI*, pages 259–274, 2020. 8

[25] So Yeon Min, Devendra Singh Chaplot, Pradeep Kumar Ravikumar, Yonatan Bisk, and Ruslan Salakhutdinov. FILM: Following instructions in language with modular methods. In *International Conference on Learning Representations*, 2022. 2, 3, 5, 6, 8, 12

[26] Alexander Pashevich, Cordelia Schmid, and Chen Sun. Episodic Transformer for Vision-and-Language Navigation. In *ICCV*, 2021. 6, 8

[27] Ethan Perez, Douwe Kiela, and Kyunghyun Cho. True few-shot learning with language models. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. 2, 3, 4, 8

[28] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. 12

[29] Timo Schick and Hinrich Schütze. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, 2021. 2

[30] Dhruv Shah, Błażej Osiński, Sergey Levine, et al. Robotic navigation with large pre-trained models of language, vision, and action. In *6th Annual Conference on Robot Learning*, 2022. 8

[31] Pratyusha Sharma, Antonio Torralba, and Jacob Andreas. Skill induction and planning with latent language. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1713–1726, Dublin, Ireland, May 2022. Association for Computational Linguistics. 1, 3, 8, 12

[32] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2, 3, 5, 8

[33] Chan Hee Song, Jihyung Kil, Tai-Yu Pan, Brian M. Sadler, Wei-Lun Chao, and Yu Su. One step at a time: Long-horizon vision-and-language navigation with milestones. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15482–15491, June 2022. 3, 6

[34] Alessandro Suglia, Qiaozi Gao, Jesse Thomason, Govind Thattai, and Gaurav Sukhatme. Embodied bert: A transformer model for embodied, language-guided visual task completion, 2021. 8

[35] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999. 1, 3

[36] Peter West, Chandra Bhagavatula, Jack Hessel, Jena Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. Symbolic knowledge distillation: from general language models to commonsense models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4602–4625, Seattle, United

States, July 2022. Association for Computational Linguistics. 2

[37] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, Oct. 2020. Association for Computational Linguistics. 6

[38] Yichi Zhang and Joyce Chai. Hierarchical task learning from language instructions with unified transformers and self-monitoring. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4202–4213, Online, Aug. 2021. Association for Computational Linguistics. 6

[39] Kai Zheng, KAI-QING Zhou, Jing Gu, Yue Fan, Jialu Wang, Zonglin Li, Xuehai He, and Xin Eric Wang. Jarvis: A neuro-symbolic commonsense reasoning framework for conversational embodied agents. *ArXiv*, abs/2208.13266, 2022. 8

Appendices

In this supplementary material, we present additional details and clarifications that are omitted in the main text due to space constraints.

- Appendix A: Implementation details and clarification of baseline models included in the main result table (cf. section 4.3 in the main paper)
- Appendix B: Prompt design choices and prompt selection under true few-shot setting (cf. section 3.2 in the main paper)
- Appendix C: Additional fine-grained analyses (cf. section 4 in the main paper)

A. Discussion and Implementation Details of Baseline Models

HLSM [3] and FILM [25] are the main baselines we compare LLM-Planner with under the few-shot setting, which adopt a similar hierarchical planning model and achieve near state-of-the-art performance on the full ALFRED data. We first briefly describe their method and how we adapt them to the few-shot setting. We also provide description of another recent method, $(SL)^3$ [31], that also operates under the low-data regime. However, it needs to use 10% of ALFRED's training data, which compares to our few-shot setting with only 100 training examples (less than 0.5%).

A.1. HLSM

HLSM [3] consists of three components: a semantic voxel map, a high-level planner, and a low-level planner. First, a 3D semantic voxel map is constructed by applying semantic segmentation and depth estimation to the visual inputs, which stores the agent's and the objects' real-time locations. Next, the high-level planner takes the language instructions, the semantic map encoding, and the previous subgoal history to predict the next subgoal. Lastly, the low-level planner is a mixture of deterministic algorithms and learned components (e.g., learning a yaw and pitch angle to face the object). HLSM first processes the sensory image input to create/update a map, which is used as an input to the high-level planner along with the language instructions to predict the next subgoal. Finally, the low-level planner maps the subgoal into a sequence of primitive actions.

To adapt HLSM to the few-shot setting, we need to re-train the components of the model that need paired trajectory-instruction data for training. For HLSM, paired data was only used for training the high-level controller. Therefore, we re-train the high-level controller with the same 100 training examples we use for LLM-Planner. Specifically, we use the same set of hyperparameters as HLSM and train the model for 6 epochs following the author. While the original HLSM focuses on the goal instruction only setting, we found that the step-by-step instructions are essential for the few-shot setting, so we concatenate goal instruction with step-by-step instructions for re-training HLSM's high-level planner. We leave the other components intact, which are downloaded from the official codebase.[2]

A.2. FILM

FILM [25] consists of four components: a semantic map, a semantic search policy, a template-based high-level planner, and a low-level planner. At the beginning of each task, five separate BERT-based classifiers [7] are used to predict five parameters (task type, target objects, receptacles, parent objects, and whether slicing is needed), each of which takes the goal and optionally the step-by-step instructions as inputs to predict the respective parameter. FILM then generates the high-level plan by choosing a pre-defined template based on the predicted task type and filling in the other parameters into the template. In addition, the semantic map is updated at each time step with the sensory image inputs. At every 25 steps, the semantic search policy predicts the coordinates of the target object on the semantic map, which are then used by a deterministic low-level planner to decide on a low-level plan to navigate from the current location to the target object's location.

Only the BERT-based classifiers need language-related data for training. Therefore, to adapt FILM to the few-shot setting, the five BERT-based classifiers are re-trained with the same 100 training examples used by LLM-Planner. Similar to HLSM, we concatenate the goal and the step-by-step instructions as input to the BERT-based classifiers. Hyper-parameters for BERT models are the default in code or the number in the paper. We use the predictions from these models to generate the high-level plans with the same pre-defined templates in FILM. We leave other components intact, which are downloaded from the official codebase.[3]

A.3. $(SL)^3$

$(SL)^3$ [31] is an another recent hierarchical planning model. $(SL)^3$ randomly samples 10% of ALFRED's training data for training. The high-level planner is based on a pre-trained T5-small [28] model, which is fine-tuned to generate high-level plans from the goal instruction. The low-level planner is another fine-tuned T5-small model, which is tasked of generating a low-level plan for each subgoal in the high-level plan. Both goal and step-by-step instructions are needed for training, but only goal instructions are needed at inference time.

[2] https://github.com/valtsblukis/hlsm
[3] https://github.com/soyeonm/FILM

Options	Task Introduction	Goal Instruction	Step-by-step Instructions	Plan List	Object List	Retrieval Message
Default	Create a high-level plan for completing a household task using the allowed actions and visible objects. Allowed actions are [action list]	Task description: [goal instruction]	Step-by-step instructions: [instructions]	(Completed, Next) plan: [subgoals]	Visible objects are [objects]	Next plan:
Punctuation	("PickupObject") (PickupObject) **PickupObject**			("PickupObject", "Apple") (PickupObject, Apple) **PickupObject, Apple**		
Naturalization	**PickupObject** Pickup Pick up			**PickupObject** Pickup Pick up		
Delimiter			**Pick up, go to** Pick up. Go to. Pick up \n Go to	**Pickup, Navigate** Pickup. Navigate Pickup \n Navigate	**Apple, orange** Apple. orange Apple \n Orange	

Table 3. For each element in our prompt design, we list the default phrasing. For the representation of actions, objects, and lists, we additionally experiment with different choices of punctuation, naturalization, and the delimiter between elements in a list. We select the optimal prompt design using LOOCV on the 100 training examples. The chosen options are highlighted in bold.

We could not compare $(SL)^3$ under the same few-shot setting as LLM-Planner because its code was not publicly available at the time of submission. However, we would like to highlight that our method achieves comparable performance on the validation set despite using only less than $1/20$ of training data than $(SL)^3$ (0.5% vs. 10% of ALFRED's training data).

A.4. Other Baselines

For other baselines included in Table 1 of the main paper, we retrieve the results directly from the published version of the corresponding paper. If the leaderboard[4] entry is better than the numbers in the original paper, we report the higher.

B. Prompt Design Choices

In-context learning with GPT-3 could be sensitive to the prompt design. In Table 3, we show different prompt design choices we have experimented for LLM-Planner. We structure our prompt into six consecutive parts: task introduction, goal instruction, step-by-step instruction, plan list, object list, and retrieval message. For each part, we have a default phrase and a list of additional options to try on top of the default phrasing signified as []. All the options listed only modify the phrase that goes in []. First, we try adding punctuation marks around actions and object. Next, we naturalize each action name as a plain English text. Lastly, we experiment with finding the optimal delimiter between action list and step-by-step instruction list. We compared comma, period, and newline inserted between the sentences. The best prompt was chosen from the LOOCV accuracy for high-level plans and is bolded.

[4]https://leaderboard.allenai.org/alfred/submissions/public

C. Additional Fine-Grained Analyses

C.1. HLP Accuracy by Task Type

We show LLM-Planner's high-level planning (HLP) accuracy breakdown by task type in Table 4. Because it is difficulty to determine a single value for the HLP accuracy for dynamic LLM-Planner, here we focus on the static version, but the HLP accuracy of the dynamic version generally correlates well with that of the static version. From the results, we observe that the results do not depend much on the difficulty of the task. For example, the task *"Stack & Place"* is often considered as the most difficult task based on the success rate of state-of-the-art models, but LLM-Planner's HLP accuracy is similar to those of easier tasks such as *"Place two"*. We find that LLM-Planner is not overly sensitive to the complexity of tasks. This suggests that it could generalize well to different types of tasks with only a few in-context examples.

Task Type	HLP Accuracy	
	Valid Unseen	Valid Seen
Pick & Place	51	46
Stack & Place	38	25
Place Two	39	45
Examine	44.4	49
Heat & Place	36	48
Cool & Place	43	46
Clean & Place	48.8	32

Table 4. Static LLM-Planner's high-level planning accuracy breakdown by task type.

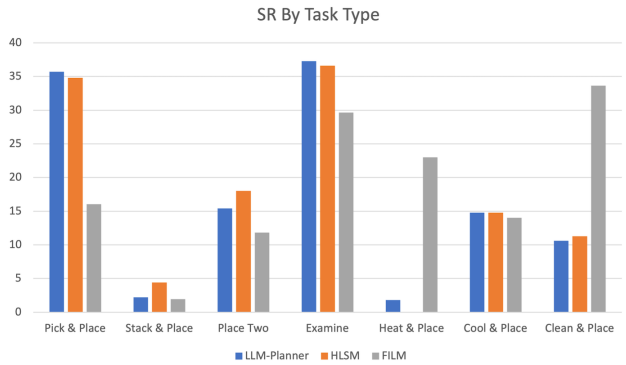

Figure 5. Success rate by task type on ALFRED valid unseen split.

C.2. End-to-End Performance by Task Type

We show the end-to-end performance breakdown by task type of dynamic LLM-Planner + HLSM in Figure 5. As a reference, we also compare with HLSM and FILM trained with the full training set of ALFRED. Keep in mind that this is not apples-to-apples comparison because LLM-Planner is under the few-shot setting. Despite that, we can see that LLM-Planner + HLSM achieves comparable performance with HLSM, and the distribution of the two are similar. This is likely due to the shared low-level planner and object detector, which introduce a similar error profile. This again shows that our few-shot high-level planner is as good as HLSM's high-level planner that is trained with the full training set. On the other hand, it also shows that there is still a large room to improve by using better low-level planners and object detectors. For example, even though our HLP accuracy for *"Heat & Place"* is 36% as shown in Table 4, we could only get 1.8% success rate due to the object detector from HLSM often failing to detect the *"microwave"*. If we use FILM's low-level planner and object detector, we may be able to achieve much better performance on this task.