2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Learning Affordances at Inference-Time for Vision-Language-Action Models

Ameesh Shah[1], William Chen[1], Adwait Godbole[1], Federico Mora[1],
Sanjit A. Seshia[1], Sergey Levine[1,2]
https://liten-vla.github.io/

***Abstract*— Solving complex real-world control tasks often takes multiple tries: if we fail at first, we reflect on what went wrong, and change our strategy accordingly to avoid making the same mistake. In robotics, *Vision-Language-Action* models (VLAs) offer a promising path towards solving complex control tasks, but lack the ability to contextually and dynamically readjust behavior when they fail to accomplish a task. In this work, we introduce *Learning from Inference-Time Execution* (LITEN), which connects a VLA low-level policy to a high-level VLM that conditions on past experiences by including them in-context, allowing it to learn the affordances and capabilities of the low-level VLA. Our approach iterates between a *reasoning* phase that generates and executes plans for the low-level VLA, and an *assessment* phase that reflects on the resulting execution and draws useful conclusions to be included in future reasoning contexts. Unlike similar approaches to self-refinement in non-robotics domains, LITEN must reflect on unstructured real-world robot trajectories (e.g., raw videos), which requires structured guiderails during assessment. Our experimental results demonstrate LITEN is able to effectively learn from past experience to generate plans that use high-affordance instructions to accomplish long-horizon tasks.**

I. Introduction

Robotic foundation models based on powerful pre-trained vision-language model (VLM) backbones have the potential to combine both the semantic and common-sense problem-solving abilities of LLMs and the flexible and dexterous end-to-end control capabilities of learned policies [1], [2], [3], [4], [5]. However, current robotic foundation models, most notably *Vision-Language-Action* models (VLAs), have primarily been studied in "single shot" settings, where they are evaluated on their ability to follow individual user commands. A practical robotic system needs to also plan through complex behaviors and, perhaps most importantly, adjust its behavior based on context and perceived capabilities. For example, if the robot needs to open a latched container, it might try to unlatch it in a particular way, and if that fails, it should modify its strategy and try a different approach. This kind of in-context adaptation has been observed as an *emergent* behavior in LLMs [6], [7], [8], but has proven difficult to enable in the robotics domain with current VLAs.

In this paper, we propose an inference-time learning method that enables robotic policies to reason through complex tasks, form plans, and then adjust those plans based on observed outcomes (without any additional training). Our method, **Learning from Inference-Time Execution (LITEN)**, employs an off-the-shelf VLM to first attempt a task by controlling a low-level VLA, then retrospectively reason about the induced robot behaviors. These insights can then be included in-context in future attempts, allowing the VLM to progressively learns *affordances* [9]: what the robot can or cannot do, subject to the constraints of dynamics, the robot embodiment, and the policy's learned behaviors. LITEN's high-level VLM "feels out" the policy's capabilities, gradually strengthening its interface with the low-level controller and *improving its high-level task reasoning as experience is accumulated*. We illustrate how LITEN can utilize past experiences in Figure 1.

Our approach can be seen as an adaptation of inference-time self-refinement approaches for agentic foundation models [8], [10], [11] to real-world robotics. As shown in Figure 2, LITEN adopts a two-phase iterative approach to self-refinement: a *reasoning* phase, followed by an *assessment* phase. In the initial reasoning phase, our high-level VLM has no prior experience of how the VLA will act when instructed, and must "guess" a plan (i.e., a sequence of subtask instructions). After the plan is attempted in the physical world, we move to the assessment phase and enlist a VLM *judge* to evaluate the result of the VLA's execution. Unlike existing self-refinement methods that can leverage precise feedback in simulated or computational environments, we must deal with unstructured data (e.g., raw videos) and draw meaningful conclusions for use in refining robot task reasonings.

To gain valuable takeaways from unstructured videos of robot executions, LITEN breaks the process of assessment down into a sequence of steps that are individually manageable by the VLM judge. Specifically, the judge assesses the outcomes of robot rollouts by determining which subtask commands failed, what the policy did incorrectly, possible reasons why, and how these errors might be avoided. These insights are then provided in-context when the VLM next attempts the task, allowing it to revise its task reasoning in light of its prior experience.

In summary, our contribution is LITEN, a novel inference-time technique for high-level VLMs to solve complex robotic tasks by learning affordances through repeated interactions with the environment, which is leveraged for better task reasoning. Crucially, LITEN requires no additional training and can be used by off-the-shelf VLMs. We demonstrate the efficacy of LITEN on a set of long-horizon manipulation tasks using the DROID Franka robot setup [12]. Our results

[1]UC Berkeley [2] Physical Intelligence. Email: ameesh@berkeley.edu.

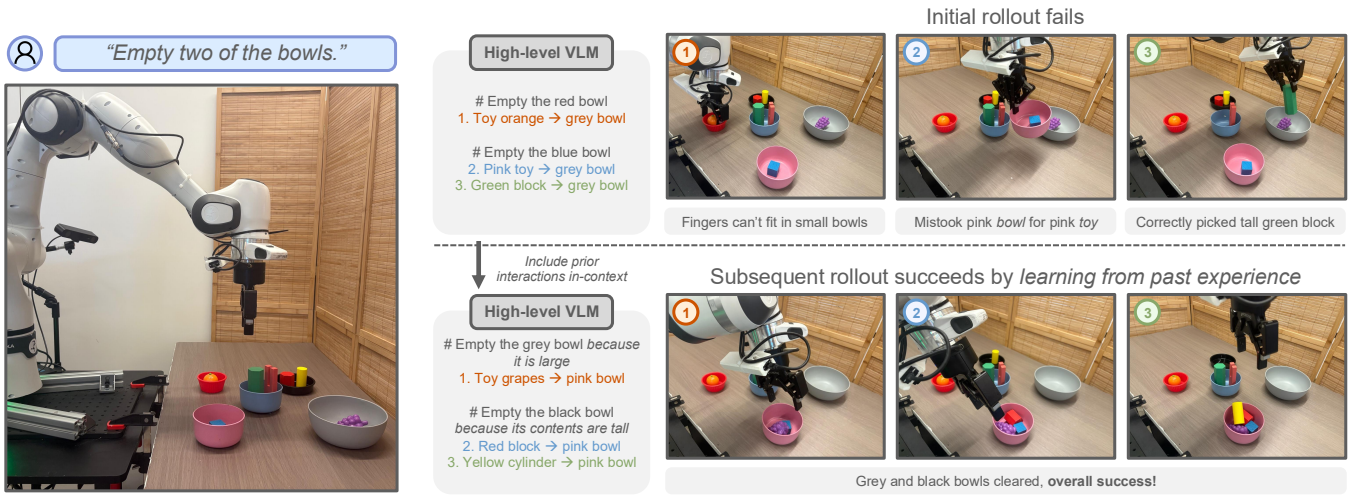

Fig. 1. Our method, *Learning from Inference Time Execution* (LITEN), is a novel approach that leverages a two-level hierarchical model to reason about complex tasks based on prior experience collected at inference-time, allowing it to learn the robot's affordances and gradually improve performance.

show that our approach is able to iteratively improve its planning capabilities from repeated attempts at a task, outperforming baseline approaches to inference-time learning that are not designed for real-world robotics.

II. Related Work

Vision-Language-Action models. Vision-Language Models (VLMs) are a powerful source of representations for control. They are popularly leveraged via behavioral cloning (BC), which yields Vision-Language-Action policies (VLAs): VLMs that have been fine-tuned on vast robot demonstration corpora [13], [14], [12], [15] into robot policies [2], [4], [3], [16]. VLAs are often generalists: due to their diverse training data, they can perform many tasks in unstructured settings, typically as commanded in text. Despite this, current VLAs are mainly trained on low-level tasks, and thus fail at high-level tasks that demand reasoning or common sense. While hierarchical [17], [18] and reasoning [19], [20], [21], [22] approaches alleviate this, such methods require expensive data collection, annotation, and training. In contrast, LITEN uses an off-the-shelf VLM to learn from inference-time rollouts by storing and reasoning about them in-context, without training.

Planning and affordance reasoning. An alternative approach that addresses this shortcoming is to use pre-trained foundation models as high-level planners or reasoners, typically interfacing with a separate low-level policy or controller [23], [24]. These methods are *zero-shot*: they forgo robot-specific training and instead use the common-sense knowledge and reasoning capabilities in VLMs to determine (1) what the robot *should* do (e.g., by decomposing a goal into subtasks) and (2) what the robot *can* do, grounded in the scene and the robot's capabilities (also called "affordances" [9]). This dichotomy is outlined by SayCan [25], which uses a language model to determine appropriate subtasks, while a separate learned value function estimates affordances. Other works learn a value function estimation for foundation-model based approaches either in an offline reinforcement learning setting [26] or through visual heuristics [27]. In contrast, VoxPoser [28] leverages (V)LMs to determine affordances by constructing a semantic voxel map in code. Other approaches couple affordance estimation and task reasoning more tightly, using the VLM both to interpret the scene and decide on appropriate behaviors [29], [30].

However, contrasting LITEN, the zero-shot nature of these approaches can also be a drawback: because the high-level VLM is not trained on robot data, it has limited understanding of the robot's capabilities and limitations, leading to suboptimal performance [25]. This is only exacerbated as considered tasks become more complex, necessitating both more flexible low-level controllers and the VLM to have a more fine-grained grasp of possible behaviors.

Inference-time learning for agentic tasks. Outside of robotics, iterative inference-time refinement methods have been explored for agentic tasks in other domains [8], [10], [11]. Such works often have similar considerations as in robotics, especially when applied to control in simulated environments [31], [32], [33]. However, translating these methods directly to real-world robot control can be difficult. Specifically, existing works (1) abstract away low-level control by delegating it to a powerful planner (often leveraging privileged simulation data), and (2) makes use of that same ground-truth simulator state for expressing agent observations, skills, and even interaction feedback in text. Both these assumptions cannot be made for real-world robotics. We compare LITEN with a direct adaptation of Reflexion [8] to illustrate our method's comparative efficacy in unstructured real-world robot manipulation applications.

III. Problem Statement

We consider the regime of robotic control for long-horizon instruction following. We define a *task* ℓ as a natural language description of the desired instruction, e.g., "Empty two of the bowls" from Figure 1.

In order to accomplish free-form natural language instructions like this, we make use of language-conditioned policies $\pi^{\text{low}}(a \mid o, \ell')$, which map observations o and input instruction text ℓ' to a distribution over low-level robot

actions a. We specifically use *vision-language-action* models (VLAs) as π^{low}, as they are an effective end-to-end approach for learning generalist language-conditioned control policies [2], [3], [4].

Critically, while VLAs are open-vocabulary policies and can condition on arbitrary language, they often struggle when given open-ended, long-horizon instructions. Instead, they excel at the shorter-horizon, atomic behaviors found in their training data. To accomplish long-horizon tasks, they also need separate high-level policies to break the overall task into behaviors that the VLA is trained for [17], [18]. To address this limitation, we use a highly-capable off-the-shelf VLM for high-level reasoning, π^{high}, decomposing the overall task ℓ into a sequence of subtask instructions $\ell'_0 \dots \ell'_n$. The VLA can then execute each subtask in sequence. When rolled out via a standard action-perception loop, this yields a subtask trajectory $\tau_i = \{(o_0, a_0)_i, (o_1, a_1)_i, \dots (o_T, a_T)_i\}$ (where actions are sampled from the VLA conditioned on the current observation and subtask language $a_t \sim \pi^{\text{low}}(\cdot \mid o_t, \ell'_i)$ and T is the length of the subtask trajectory). The overall trajectory is simply the concatenation of all these subtask trajectories τ_i in sequential order. The overall task ℓ is successful if it is accomplished by the end of the last subtask trajectory.

Finally, we note that the VLA has learned to map commands to a diverse range of low-level actions. While we have a sense of what language the VLA has been trained to follow (which is useful for giving examples of "reasonable" subtask language ℓ'_i for π^{high} to output), we do not have any a priori description of its affordances (e.g., in what scenarios it is able to accomplish certain commands, and with what likelihood). We thus consider an *iterative multi-round setting*, wherein the system can attempt the task multiple times. Critically, the system can pass forward some information to subsequent attempts. Our experiments thus compare different ways to use pre-trained VLMs as π^{high}, specifically showing that our approach, LITEN, is an effective way to infer affordances and improve performance from past robot rollouts.

IV. Learning from Inference-Time Execution

We now present LITEN, an approach that allows the VLM planner π^{high} to learn the affordances of the VLA π^{low} at inference time through repeated interactions with the physical world, subsequently using these affordances to improve its planning capabilities. We show that, by reasoning about prior attempts at a task – how they succeed or fail, the underlying semantic or physical constraints that lead to these outcomes, and what sorts of behaviors the policy is empirically suited to – LITEN is an effective way to gradually improve task success rates as more experiences are accumulated in-context, all *with no extra policy training*.

LITEN consists of an iterative loop with two phases: a *reasoning* phase and an *assessment* phase. In the reasoning phase (Section IV-A), the VLM *reasoner* is given a task instruction ℓ and an initial image observation of the environment. It is instructed to reason through and decompose this goal into subtasks, which the VLA then executes as described in Section III. We then move to the assessment phase (Section IV-B), where we evaluate the outcome of each subtask execution in a structured format, invoking a VLM *judge* with a chain of prompts to draw meaningful conclusions from unstructured trajectories. Finally, the result of this assessment is then added back in-context to the prompt for the VLM reasoner in the next iteration, which generates a new plan that takes the prior experience into account (Section IV-C). We provide a diagrammatic representation of LITEN in Figure 2. We now detail each phase of LITEN.

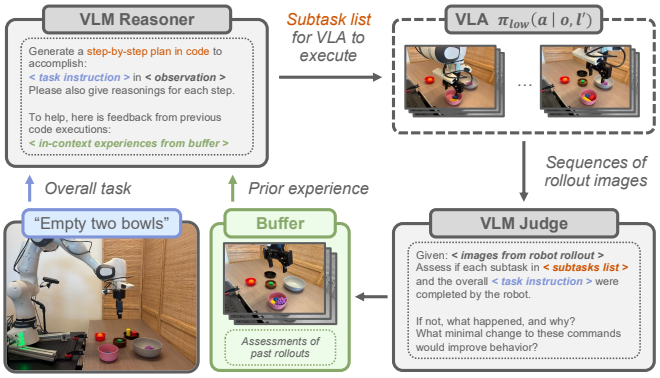

Fig. 2. An overview of our approach. LITEN cycles between **(1) the reasoning phase**, wherein the VLM reasons about the task to decompose the task into subtasks that the VLA low-level policy can roll out in sequence and **(2) the assessment phase**, wherein the VLM judges the outcomes of inference-time rollouts. The reasoning phase includes the outputs of the assessment phase by including them in the VLM's context, allowing the reasoner to iteratively learn robot affordances and improve performance.

A. Reasoning Phase: Generating and executing plans

The reasoning phase begins by asking a VLM to act as π^{high}. We give it the overall task ℓ and initial observation image, then prompt it to produce a list of subtasks that solve this task, along with a justification for each step.

To help direct π^{high}'s outputs, we supply the VLM reasoner with additional context. First, we give it provide a description of language commands that are representative of instructions in the VLA's pre-training data. This constrains π^{high}'s outputs to the "style" of language subtasks that π^{low} can follow (though critically *not* its affordances, i.e., when each command is feasible). We also describe the general class of instructions that are relevant to our task setup: in our setting, these are pick-and-place and move operations by a robot arm. Lastly, to ground the reasoner in its environment, we employ a similar technique to prior work [34], [35] and provide it with a list of manipulable objects, identified by a separate VLM call. Executing each subtask ℓ'_i in sequence with the VLA yields a corresponding trajectory segment $\tau_i = \{(o_0, a_0)_i, (o_1, a_1)_i, ...\}$.

B. Assessment Phase: Learning from previous executions

Once a plan has been fully executed, we begin the assessment phase by collecting the sequence of sub-trajectories for each subtask attempted by π^{low}, i.e., $\{(\ell'_0, \tau_0), \dots (\ell'_n, \tau_n)\}$. To generate useful feedback for future iterations, we present a *structured assessment procedure* that asks a VLM judge to

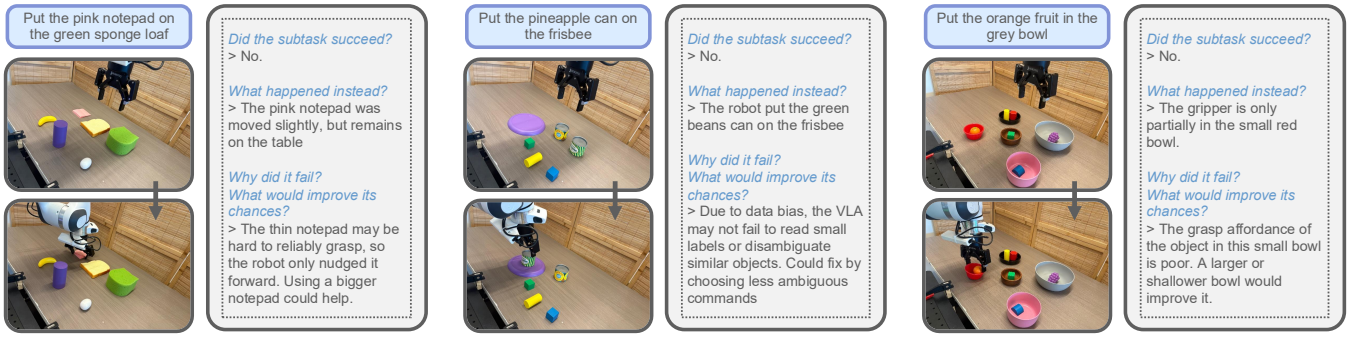

Fig. 3. Examples of VLM judge's prompt and output for three subtasks (abridged for length). These generations are included in-context for the VLM reasoner in subsequent iterations attempting the task. Additional full examples are available in our codebase at https://github.com/ameesh-shah/liten-vla

answer a chain of prompts about each subtask that mirrors human reasoning. The prompts are as follows:

1) *Did it succeed?* We give a subtask instruction ℓ'_i and the first and last observation images of τ_i as input to the judge, and ask whether or not the robot successfully accomplished ℓ'_i.
2) *What happened instead?* In the case of failure, we ask the judge to describe what the robot did in the environment instead of ℓ'_i. In this step, we experimented with providing a video (sampled at various frame rates) of τ_i but found that our VLMs struggled to accurately comprehend unstructured sub-trajectory videos. For our experiments, we found it sufficient to provide the first and last observation images when asking the judge to describe what happened.
3) *Reason about failure.* We give the intended subtask ℓ'_i, the description of what actually happened from the previous step, and the initial observation image to the judge and ask it to reason about why π^{low} failed in the way it did. To ensure that the judge can correctly diagnose failure causes, we include substantial context in our prompt that outlines a general VLA training and inference procedure and briefly describes the way π^{low} attends to language (e.g., 'Our VLA tends to pay attention to the color, shape, and spatial orientation of objects in the instruction.') We additionally ask the VLM to suggest what *minimal* changes to either ℓ'_i or the environment would improve the chance of success. We emphasize minimality to encourage suggested changes that are grounded in the specific task context.

We provide an example illustrating the above steps in Figure 3. In the case of a successful subtask, we instead ask a VLM to briefly describe the environment in which the success occurred. Once we have generated assessments for each subtask, we then ask a VLM to evaluate the overall task. We provide the structured assessments for all subtasks in an execution, along with overall task language ℓ, and ask the judge to describe why the overall task succeeded or failed based on the outcomes of each subtask.

C. Using feedback in future iterations

Upon the start of the next reasoning phase, the structured feedback generated in all previous assessment phases is added in-context to the plan generation prompt. To ensure that assessments are used effectively, we add instructions on how to use prior experience to the VLM reasoner's prompt. In the prompt, we emphasize that a subtask's success is dependent on the environment, and that subtasks with multiple occurrences in-context do not necessarily share the same environment configuration with one another nor the current environment. Our prompt also notes that the VLA is stochastic, and its performance on the same subtask may vary across executions. We instruct the VLM reasoner to prioritize using subtask instructions in the following order: (1) instructions that are very likely to succeed in the current environment based on prior experience, (2) instructions that have not yet been tried, and (3) instructions that are unlikely to succeed but have been rephrased in a way that maximizes their success likelihood. The VLM reasoner takes into account its prior experience and the aforementioned instructions, then generates a new plan for execution.

D. Implementation

We now describe our concrete instantiation of LITEN on a real-world robotics setup. We use GPT-5-mini [36] as our high-level VLM in both the reasoning and assessment phases, given its cost effectiveness and state-of-the-art reasoning capabilities. For our low-level policy, we use $\pi_{0.5}$-DROID [17], an open-source state-of-the-art VLA that has been fine-tuned on the DROID dataset [12]. Accordingly, our experiments use the standard DROID robot setup: a tabletop with a 7-DoF Franka Emika Panda robot arm and a 2F-85 Robotiq gripper end-effector operating at 5Hz.

The reasoning phase takes as input a user-specified ℓ, and an initial image of the scene that comes from a fixed ZED 2.0 camera on the right side of the tabletop. The reasoning phase is a single VLM request that includes the following in addition to the aforementioned inputs: the top-level plan generation prompt for ℓ, the in-context experience collected from prior attempts, and the instructions on how to use past experience. The VLM generates Python code with comments explaining its reasoning to serve as the plan and uses a

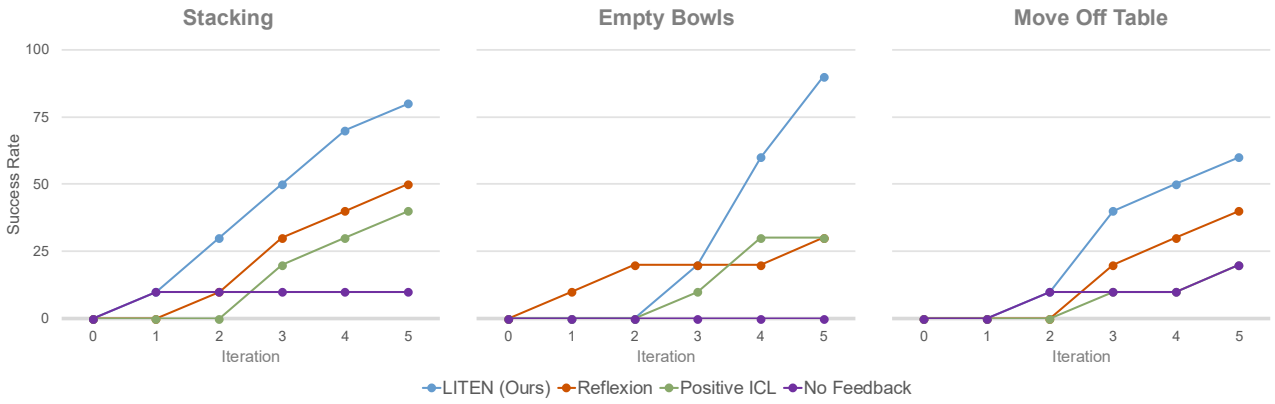

Fig. 4. Success rates for the full completion of each multi-stage task over five attempt iterations. Our results show that LITEN is able to make effective use of prior attempts and consistently improves its plans as more experience is collected. Results are averaged over 10 trials.

provided API endpoint that allows it to invoke the VLA for a subtask ℓ_i'. Additionally, we ask the VLM reasoner to justify its subtask generations in natural language.

During execution, $\pi_{0.5}$-DROID receives joint angle proprioception and RGB images from the same right-side ZED camera and a ZED mini wrist camera. Subtasks are run with a maximum horizon of 300 time steps. $\pi_{0.5}$-DROID produces joint velocity action chunks [37] of length 8, so the VLA only runs inference every 8 steps. The robot's position is reset to home between subtasks.

In the assessment phase, the initial and final image observations from each subtask are provided to the chain of prompts in our structured assessment template. Each step outlined in Section IV-B is a separate VLM request that uses the output from the previous steps. The results of all subtask assessments are then provided in a single request to the VLM judge when assessing the overall task. The overall task and subtask assessments, including the images used in those assessments, are then stored and provided in a hierarchical structure (subtasks are coupled with their overall task) in-context to future reasoning phases. Our implementation of LITEN, including the full prompts used, is available in our codebase.

V. Experiments

We evaluate LITEN on a collection of challenging multi-step tasks that requires the high-level VLM to learn relevant affordances, including (1) what the low-level robot controller is capable of and (2) spatial and physical properties of the environment and embodiment. Through our experiments, we seek to answer the following questions:

- **RQ.1.** Can LITEN learn task-relevant affordances in its environment by interacting with the physical world?
- **RQ.2.** As it gains experience, does LITEN effectively learn from both successes and failures, and iteratively improve its plans for complex tasks?
- **RQ.3.** What is the importance of each step in the assessment phase of LITEN in facilitating learning from past experiences?

A. Experimental Setup

We select three complex, multi-stage tasks that are achievable by composing behaviors and instructions akin to those found in the DROID dataset. To ensure that our VLA can consistently achieve our tasks, we fine-tune $\pi_{0.5}$-DROID on a small number of collected demonstrations to help it adapt to our specific tabletop and task setups. We collect 150 total demonstrations per multi-stage task, which are divided up into 5-15 distinct subtasks. The initial configuration of our task does not vary across iterations in a given trial, but does vary slightly across trials; e.g., some object initial positions are switched. Our tasks are depicted in Figure 5 and described as follows:

1) **Stacking.** The robot must stack a total of three objects atop one another. The scene has 3 small blocks, 2 medium-sized cans, and a large upside down plate. The high-level VLM must learn which objects can be easily stacked atop one another by the VLA and which objects are too difficult to precisely balance.
2) **Emptying Bowls.** A number of bowls of different depths and sizes are set on the table. The robot must empty two bowls by moving their contents to other bowls. The high-level VLM must learn which bowls are either large enough for the gripper end-effector to fit in *or* shallow enough to allow objects to protrude and be grasped by the arm.
3) **Moving Off Table.** The robot must move objects on the table onto other objects such that only three objects remain in direct contact with the table. The high-level VLM must learn which objects are manipulable by the VLA, and which objects can serve as landing spots for other objects without rolling or falling off.

Baselines. We compare our approach against a number of alternative approaches to learning from inference-time experience that do not reason about experiences in the same way as LITEN. While to our knowledge prior VLA methods do not utilize inference-time experience in-context, we adapt *Reflexion* [8], an inference-time learning technique for agentic LLMs, to a robotic setting. Our adaptation 'reflects' on unstructured videos of executed plans, and uses

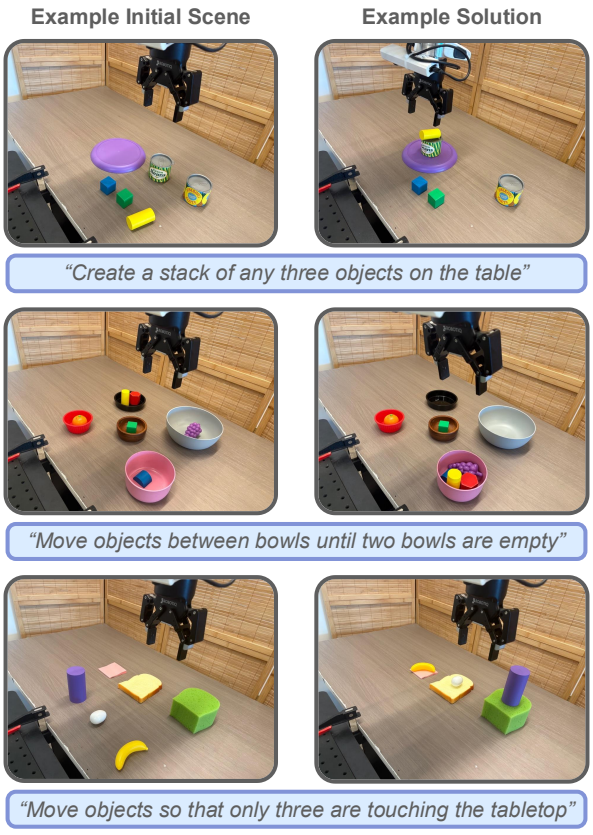

Fig. 5. Examples of initial configurations and solutions for our tasks.

these reflections in-context for subsequent iterations. This baseline can be thought of as a naïve adaptation of standard self-refinement approaches to real-world robotics. We also compare against an approach that only stores successful subtask attempts and provides those successful attempts in-context for future plan generations, called *Positive-ICL*. This approach mirrors existing methods that use positive in-context examples for zero-shot open-world robotic manipulation [29], and can be viewed as an ablation of our method that does not use negative examples. Lastly, we compare against a naïve baseline that regenerates a new plan at every iteration without using any feedback, called *No-Feedback*.

Ablations. We answer **RQ.3.** by conducting an ablation study where we remove steps from the assessment phase (see Section IV-B) and run LITEN in the same manner as described above for the Empty Bowls task. We ablate the approach in two ways: (1) removing the failure reasoning (**w/o failure reasoning**: "why did the policy fail at this subtask?") and (2) removing the outcome analysis as well (**w/o failure reasoning and outcome**: "what did the policy do instead?"), leaving only whether the subtasks were successful or not.

B. Results

Main results. In Figure 4, we show the results of each approach in accomplishing the tasks over a total of five iterations, averaged over ten trials. For each trials, we run each method for a maximum of five iterations, stopping early if the attempt is successful. The success rates represent the rate of accomplishing the entire task and do not consider partial credit. Our results show that LITEN is able to make use of its

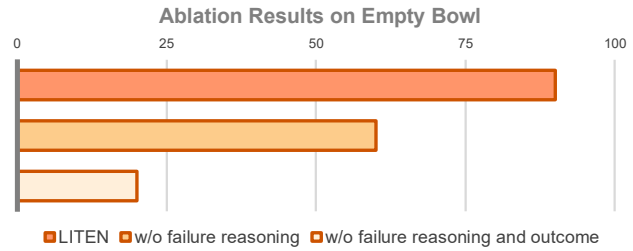

Fig. 6. Success rates after five past attempts after ablating various components of LITEN's VLM judge. When any part of the assessment phase is ablated, performance drops dramatically, indicating that prior experience is most impactful when it includes whether the past rollout succeeded or not, what the policy did instead if not, and why that might be the case.

collected experience and successfully instruct π^{low} with high-affordance subtask instructions to accomplish the overall tasks, answering **RQ.1.** and **RQ.2.** affirmatively. LITEN's success rate increases consistently over consecutive attempts, indicating that it effectively uses additional prior experience to further improve its plans. The baseline approaches are substantially less effective at leveraging prior experience. Notably, the naïve No-Feedback baseline is practically unable to accomplish any tasks, showing that the VLM will not occasionally generate a correct plan as a result of random chance. This shows the importance of learning from past experience at inference time.

Ablation results. We present the performance of LITEN under various ablation conditions in Figure 6, demonstrating the importance of the knowledge gained in each step of the assessment phase. Providing only the success or failure of each subtask (without failure reasoning or outcome analysis) yields the worst performance. Removing the failure reasoning only performs better, but will on occasion attribute differences between intended and actual outcomes to the instruction's language as opposed to physical explanations. The explicit "reason about failure" step provides useful context about why failures may have occurred, such as suggesting the limitations of VLA capabilities or physical properties of objects, and thus allows LITEN to learn more sophisticated affordances.

Qualitative analysis. We provide a qualitative analysis to answer **RQ.1.** and describe the affordances LITEN learns in our experiments. We find that LITEN is particularly successful at learning from failures that either indicate (1) biases from the VLA or (2) physical properties that cause obvious difficulties in control.

To exemplify (1), in the Stacking task, the VLA is biased towards manipulating larger objects in the scene, such as the two cans or the yellow cylinder block. In an attempt from our experiments, the high-level VLM instructed the VLA to "put the green cube on the purple plate", but the VLA instead moved the green beans can onto the purple plate. LITEN was able to clearly identify this mistake made by the VLA and recognize that moving the green can to the purple plate is a subtask that can be part of a potential solution. The subsequent plan included "put the green beans can on to the purple plate".

Regarding (2), we found that the high-level VLM often

suggested initial plans that require precise control. For the Stacking task, the high-level VLM would frequently generate a plan involving stacking the three small blocks instead of the larger cans and plate, which is comparatively more difficult and requires finer manipulation skills. In the Moving Off Table task, the VLM occasionally suggested moving the green sponge on top of other objects, despite it being the largest object on the table and too wide for the gripper. However, LITEN enabled the high-level VLM to draw meaningful conclusions from these attempts (e.g., "the VLA may lack precise top-down placement abilities when stacking the blue and green blocks" or "the gripper could not grasp the green sponge due to size constraints") and generate plans that adjusted accordingly.

The baselines are unable to learn as effectively. Because the Positive-ICL baseline only retains successful subtasks in-context, the method must effectively "get lucky" and find successful subtasks through the randomness of the VLM-generated plans, leading to high inefficiency. This illustrates how LITEN's fine-grained rollout assessments critically provide a strong signal for understanding the robot's capabilities and leveraging it for improvement. The Reflexion baseline reasons over entire raw video trajectories, which we found often led to poor comprehension of physical outcomes. For example, we found that entire-video reflections would often hallucinate objects that were not in the scene, or mistakenly assume subtasks were successfully completed. This underscores the challenge of extracting meaningful feedback from raw robot interaction data.

VI. Discussion

A. Failure Cases

We characterize the failure cases of LITEN when it fails to correctly plan for a task after multiple execution attempts. The majority of failures can be attributed to either (a) the stochasticity of the VLA, (b) LITEN's tendency to attribute control failures to the wording of a subtask, or (c) LITEN's inability to reason causally about subtask orderings.

For the first failure case, a small number of potential subtasks in our experiments have high variance, such as stacking one can atop another in the Stacking task. If LITEN experiences a successful subtask execution in an early iteration, it will tend to fixate on using this subtask, even if multiple subsequent executions of the same subtask fail.

In the second case, if a subtask execution fails due to a control error, LITEN may misdiagnose this failure as a result of imprecise language in the subtask instruction. For example, if the subtask "put the blue block on the green block" fails, but the blue block gets moved close to the green block, LITEN may attribute this failure to the language and provide an instruction like "put the blue block directly on top of the center of the green block."

The third case is most easily understood via example. In the Move Off Table task, overall success is highly sensitive to the ordering and control precision of each subtask. In one instance, the purple cylinder was successfully placed on the green sponge. However, the next subtask was placed the white egg also on the green sponge, which, while successful, knocked the purple cylinder back onto the table in the process. Although the VLM judge managed to accurately describe this phenomenon during the assessment phase ("The VLA likely ... displaced the cylinder when placing the egg"), the VLM reasoner struggled to make use of this insight in future plans.

B. Analysis and Limitations

We now take a more prescriptive view of the cases discussed earlier. Both the first and second failure cases represent failure of the VLM to adapt to VLA affordances on *individual* subtasks. In the first case, the VLM reasoner needs to strike a balance between avoiding "borderline affordances" of VLAs, while attempting to leverage the VLA's steerability through language at the same time.

The second failure case is an artifact of using language as the reasoning mode. We expect that as VLM video comprehension capabilities improve, the VLM judge can properly classify infeasible actions that no amount of subtask language modification will fix, mitigating this failure mode.

The last case represents what we see as the most salient current limitation of LITEN. Unlike the first two failure cases, it exhibits a sequence of successful individual subtasks but a failure at the overall task level. This indicates a deficiency in the ability of the VLM to reason across subtasks. Thus, instilling the high-level VLM reasoner with stronger logical and causal reasoning is a clear area of future improvement for inference-time affordance learning.

VII. Conclusion

In this work, we present LITEN, a novel inference-time learning method that enables generalist robot policies to reason about complex tasks, attempt them in the real world, and then distill insights from those past attempts to improve future reasoning, without requiring additional training. LITEN iterates through a two-phase process that first generates plans for a VLA using a high-level VLM, then assesses the execution of that plan through a chain of prompts that enables the VLM to draw meaningful conclusions for use in subsequent planning iterations. Our experimental results show that LITEN effectively uses past experience to solve complex manipulation tasks that require an understanding of the robot's affordances.

There are many exciting directions for future work. Although LITEN is only demonstrated on single tasks, its usage in a *lifelong learning* setting is compelling, where past experience across many different tasks is used to solve new tasks. To that end, we are interested in understanding what past experiences are more or less useful in efficiently gaining affordances. We are also interested in scaling up LITEN to settings with large amounts of prior data that cannot entirely be fit in-context, which will require adapting context management techniques such as RAG [38].

Acknowledgements

The authors thank Catherine Glossop, Paul Zhou, and Kevin Black for their help and feedback. This work was

supported in part by DARPA Contract FA8750-23-C-0080 (ANSR) and the DARPA Contract HR00112490425 (TIAMAT), by Berkeley Deep Drive, and by Nissan and Toyota under the iCyPhy center.

REFERENCES

[1] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[2] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[3] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, *et al.*, "π_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

[4] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, *et al.*, "Openvla: An open-source vision-language-action model," in *Conference on Robot Learning, 6-9 November 2024, Munich, Germany*, ser. Proceedings of Machine Learning Research, P. Agrawal, O. Kroemer, and W. Burgard, Eds., vol. 270. PMLR, 2024, pp. 2679–2713.

[5] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, D. Sadigh, C. Finn, and S. Levine, "Octo: An open-source generalist robot policy," https://octo-models.github.io, 2023.

[6] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, *et al.*, "Self-refine: Iterative refinement with self-feedback," *Advances in Neural Information Processing Systems*, vol. 36, pp. 46 534–46 594, 2023.

[7] Z. Gou, Z. Shao, Y. Gong, Y. Shen, Y. Yang, N. Duan, and W. Chen, "Critic: Large language models can self-correct with tool-interactive critiquing," *arXiv preprint arXiv:2305.11738*, 2023.

[8] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 8634–8652, 2023.

[9] J. J. Gibson, *The Ecological Approach to Visual Perception: Classic Edition*. Psychology Press, 2014.

[10] Y. Tong, D. Li, S. Wang, Y. Wang, F. Teng, and J. Shang, "Can LLMs learn from previous mistakes? investigating LLMs' errors to boost for reasoning," *arXiv preprint arXiv:2403.20046*, 2024.

[11] S. An, Z. Ma, Z. Lin, N. Zheng, J.-G. Lou, and W. Chen, "Learning from mistakes makes LLM better reasoner," *arXiv preprint arXiv:2310.20689*, 2023.

[12] A. Khazatsky, K. Pertsch, *et al.*, "DROID: A large-scale in-the-wild robot manipulation dataset," in *Robotics: Science and Systems XX, Delft, The Netherlands, July 15-19, 2024*, D. Kulic, G. Venture, K. E. Bekris, and E. Coronado, Eds., 2024. [Online]. Available: https://doi.org/10.15607/RSS.2024.XX.120

[13] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, *et al.*, "Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 6892–6903.

[14] H. R. Walke, K. Black, T. Z. Zhao, Q. Vuong, *et al.*, "Bridgedata V2: A dataset for robot learning at scale," in *Conference on Robot Learning, CoRL 2023, 6-9 November 2023, Atlanta, GA, USA*, ser. Proceedings of Machine Learning Research, J. Tan, M. Toussaint, and K. Darvish, Eds., vol. 229. PMLR, 2023, pp. 1723–1736. [Online]. Available: https://proceedings.mlr.press/v229/walke23a.html

[15] Z. Jiang, Y. Xie, K. Lin, Z. Xu, W. Wan, A. Mandlekar, L. J. Fan, and Y. Zhu, "Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 16 923–16 930.

[16] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, *et al.*, "Gr00t n1: An open foundation model for generalist humanoid robots," *arXiv preprint arXiv:2503.14734*, 2025.

[17] P. Intelligence *et al.*, "$\pi_{0.5}$: a vision-language-action model with open-world generalization," 2025.

[18] L. X. Shi, B. Ichter, M. Equi, L. Ke, K. Pertsch, Q. Vuong, J. Tanner, A. Walling, H. Wang, N. Fusai, *et al.*, "Hi robot: Open-ended instruction following with hierarchical vision-language-action models," *arXiv preprint arXiv:2502.19417*, 2025.

[19] Gemini Robotics Team and others, "Gemini robotics: Bringing AI into the physical world," 2025.

[20] M. Zawalski, W. Chen, K. Pertsch, O. Mees, C. Finn, and S. Levine, "Robotic control via embodied chain-of-thought reasoning," 2024.

[21] W. Chen, S. Belkhale, S. Mirchandani, O. Mees, D. Driess, K. Pertsch, and S. Levine, "Training strategies for efficient embodied reasoning," 2025.

[22] F. Lin, R. Nai, Y. Hu, J. You, J. Zhao, and Y. Gao, "Onetwovla: A unified vision-language-action model with adaptive reasoning," *CoRR*, vol. abs/2505.11917, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2505.11917

[23] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, *et al.*, "Inner monologue: Embodied reasoning through planning with language models," *arXiv preprint arXiv:2207.05608*, 2022.

[24] A. Zeng, M. Attarian, B. Ichter, K. Choromanski, A. Wong, S. Welker, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, J. Lee, V. Vanhoucke, and P. Florence, "Socratic models: Composing zero-shot multimodal reasoning with language," 2022.

[25] B. Ichter, A. Brohan, *et al.*, "Do as I can, not as I say: Grounding language in robotic affordances," in *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, ser. Proceedings of Machine Learning Research, K. Liu, D. Kulic, and J. Ichnowski, Eds., vol. 205. PMLR, 2022, pp. 287–318.

[26] M. Nakamoto, O. Mees, A. Kumar, and S. Levine, "Steering your generalists: Improving robotic foundation models via value guidance," *arXiv preprint arXiv:2410.13816*, 2024.

[27] Y. J. Ma, J. Hejna, C. Fu, D. Shah, J. Liang, *et al.*, "Vision language models are in-context value learners," in *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*, 2025.

[28] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3D Value Maps for Robotic Manipulation with Language Models," *arXiv preprint arXiv:2307.05973*, 2023.

[29] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as Policies: Language Model Programs for Embodied Control," in *IEEE International Conference on Robotics and Automation*. IEEE, 2023, pp. 9493–9500.

[30] K. Fang, F. Liu, P. Abbeel, and S. Levine, "MOKA: Open-World Robotic Manipulation through Mark-Based Visual Prompting," *Robotics: Science and Systems (RSS)*, 2024.

[31] Y. Du, O. Watkins, Z. Wang, C. Colas, T. Darrell, P. Abbeel, A. Gupta, and J. Andreas, "Guiding pretraining in reinforcement learning with large language models," 2023.

[32] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, "Voyager: An open-ended embodied agent with large language models," *arXiv preprint arXiv:2305.16291*, 2023.

[33] V. Bhat, A. U. Kaypak, P. Krishnamurthy, R. Karri, and F. Khorrami, "Grounding llms for robot task planning using closed-loop state feedback," *arXiv preprint arXiv:2402.08546*, 2024.

[34] B. Chen, F. Xia, B. Ichter, K. Rao, K. Gopalakrishnan, M. S. Ryoo, A. Stone, and D. Kappler, "Open-vocabulary queryable scene representations for real world planning," *arXiv preprint arXiv:2209.09874*, 2022.

[35] L. Smith, A. Irpan, M. G. Arenas, S. Kirmani, D. Kalashnikov, D. Shah, and T. Xiao, "Steer: Flexible robotic manipulation via dense language grounding," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 16 517–16 524.

[36] OpenAI *et al.*, "GPT-5 System Card," *OpenAI Blog*, 2025.

[37] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," in *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, K. E. Bekris, K. Hauser, S. L. Herbert, and J. Yu, Eds., 2023. [Online]. Available: https://doi.org/10.15607/RSS.2023.XIX.016

[38] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP tasks," *Advances in neural information processing systems*, vol. 33, pp. 9459–9474, 2020.