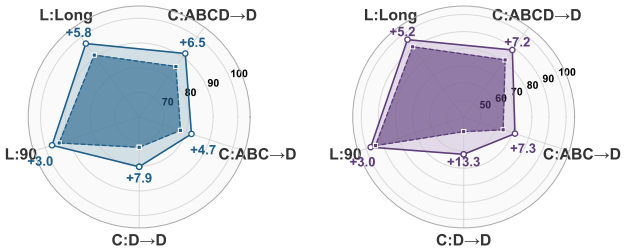
# Beyond the Moment: Conditioning Frozen VLAs on Memory for Long-Horizon Manipulation Tasks

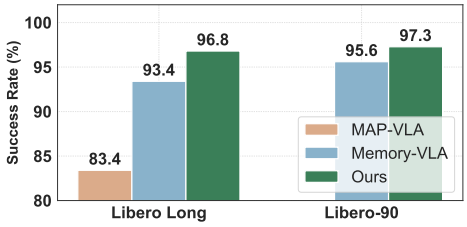
**Aditya Ramesh** [*1] **Jatin Chauhan** [*1] **Akshay Govind Srinivasan** [*1] **Shivam Bhardwaj** [1] **Manohar Kaul** [1]

## Abstract

Foundation robotics models, most popularly Vision Language Action (VLA) models, struggle to perform well over long horizon tasks due to their reliance on immediate sensory input. This induces compounding errors over inference timesteps, further exacerbated by non-robust backbones. To address this, we introduce *Training-Free Memory Conditioned Action Generation*, a non-parametric retrieval-augmented framework that conditions a frozen VLA on historical expert trajectories. Our approach constructs a memory of expert demonstrations and utilizes a state-centric retrieval mechanism to guide action generation without any fine-tuning whatsoever. By performing extensive evaluation on 5 datasets over SOTA models, we show relative gains of upto 27% on task completion success. As an additional contribution we extend the popular CALVIN benchmark to task horizon of 6 and beyond, showcasing relative gains of upto 30%, while also demonstrating robustness to corrupted observations. Real-world experiments on complex tasks further demonstrate performance gains of up to 2X.




*(a)* FLOWER VLA Comparison *(b)* MODE VLA Comparison




*(c)* Comparison with Prior Memory Frameworks

*Figure 1.* Comparison of VLA performance across *CALVIN (C)* (Level-5) and *LIBERO (L)* **(a–b)**. Radar plots highlight consistent performance of conditioning on memory (lighter shade) gains against non-memory case (shaded in dark). **(c)** The bar plot on *LIBERO* highlights gains using our memory framework against other memory augmented VLAs (which require gradient based training updates and architectural changes).

## 1. Introduction

The evolution of robotics models, specifically Vision-Language-Action (VLA) models, has fundamentally shifted the paradigm of robotic control from specialized, task-specific policies to generalist agents capable of following natural language instructions (Ma et al., 2026). By aligning large-scale visual encoders with language reasoning, current state-of-the-art VLAs effectively ground semantic commands into low-level control actions, demonstrating remarkable generalization capabilities across diverse embodiments (Kawaharazuka et al., 2025).

Despite the initial success, these models exhibit severe shortcomings (Sapkota et al., 2025). Since these models operate primarily as reactive Markovian systems (typically approximated as first order), therefore the action sequences are predicted solely based on immediate visual observations and task embeddings while neglecting the rich historical context of past successes hinders the latent capability, especially in the foundation model regime (Shi et al., 2025). This lack of explicit experience utilization substantially limits their performance in long-horizon tasks requiring complex sequential action trajectories, where temporal consistency and error recovery are critical, leading to rapidly compounding errors as shown in figure 2. Non-robust VLM backbones (Vo et al., 2025; Hancock et al., 2025) have also been of attention recently highlighting that these models struggle to distinguish high-redundancy observations, aka perceptual aliasing, without historical context and thus resulting

[1]Fujitsu Research. Correspondence to: Aditya Ramesh <Aditya.Ramesh@fujitsu.com>, Jatin Chauhan <chauhan.jatin@fujitsu.com>.

Published as a workshop paper at SCALE - ICML 2026

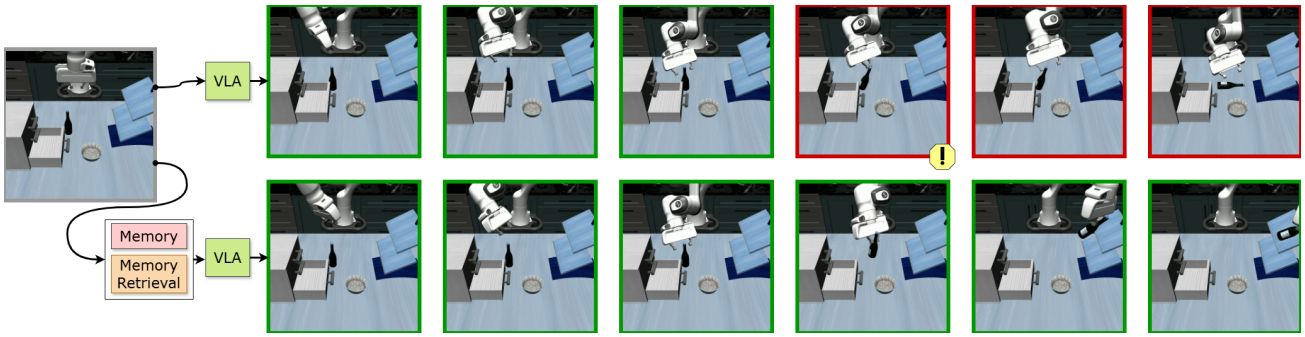



*Figure 2.* Effect of memory on long-horizon task (*Place the bottle in the rack*). **Top row:** A VLA policy without memory acting as first order Markovian process. A small drift earlier during the task (*the bottle getting knocked over*) compounds the errors over time eventually leading to task failure. **Bottom row:** Same VLA policy augmented with our memory framework leveraging memory conditioned action prediction to incorporate past experience, leading to successful task completion. Green and red borders indicate successful and failed actions, respectively.

in mode-averaging and oscillation between incompatible sub-goals. For example - visually indistinguishable pre and post grasp scenes in manipulation tasks induce perceptual aliasing and unstable action predictions.

Towards initial steps, approaches such as in-context learning for robots (Fu et al., 2025; Yin et al., 2025) have been proposed, however these approaches require extensive fine-tuning for specific architectures and cannot be applied out of the box. Other works such as MemoryVLA (Shi et al., 2025) and ExpReS-VLA (Syed et al., 2025) explore memory-augmented VLAs. However, their improvements are generally achieved through additional training and architecture-specific design choices, which can incur significant computational cost. These requirements further induce issues related to fine-tuning such as overfitting and over-reliance to specific tasks, exhaustive hyperparameter search and more (Parthasarathy et al., 2024; Barnett et al., 2024).

To circumvent these issues, we propose a *completely training-free* and *model-algorithm agnostic* memory framework which can be conditioned upon with any existing trained model directly at the inference time. The *training-free* property ensures no fine-tuning is required over the memory or the VLA policy whatsoever. This allows us to bypasses strict hardware requirements (memory construction requiring less than 40 GB on a single NVIDIA A100 for our framework, section 4). Whereas, the *model-algorithm agnostic* supports generalized plug-and-play implementation with any model based on the production requirements of user (Shi et al., 2024; Yan et al., 2024). Starting from the frozen and already trained model, we explicitly construct our (external) memory $\mathcal{M}_{ext}$ by slicing expert training trajectories into sliding-window chunks. This construction accounts for both the robot's state and the vision-language instruction embeddings computed via the frozen model. This data structure inherently enforces temporal consistency by retaining the causal evolution of the robot's state across the horizon, providing the model with the necessary episodic context to distinguish between perceptually similar but temporally distinct situations (Baddeley, 2006; Reyna & Brainerd, 1995). While our algorithmic framework also allows us to leverage arbitrary data structures and a variety of metrics to organize and retrieve from the memory, we leverage Hierarchical Navigable Small World(HNSW) (Malkov & Yashunin, 2020) as an effective high recall structure that works well in high dimensions. Our framework is visualized in figure 3. We specifically note that unlike local task-level working memory introduced in (Shi et al., 2025) and focus of benchmarks such as (Cherepanov et al., 2025), our memory framework is designed to retrieve the relevant context from the robot's historical task observations to undertake optimal actions, highlighting a different paradigm.

We validate our design through extensive experiments across 5 datasets from the most popular CALVIN and LIBERO benchmarks using two SOTA VLA models. As illustrated in figure 1a,1b, our framework yields consistent gains, effectively expanding the performance envelope of base policies. Despite the challenges highlighted against prior memory architectures, we still observe consistent gains over previous works such as (Shi et al., 2025; Li et al., 2025c) as shown in figure 1c. Our empirical analysis (section D.2) further confirms that conditioning on memory reduces failure rates by up to 92%. To summarize, our contributions are:

1. **Novel Framework:** To the best of our knowledge, we propose the first completely training-free and model-agnostic memory module that can be plugged into any VLA policy at inference time (section 3).

2. **Adaptive Retrieval & Fallback:** We adapt state-centric retrieval mechanism that efficiently isolates relevant temporal history (section 3.2.2), integrated with an adaptive memory fallback strategy (section 3.2.5) that is conditionally activated to generate actions (sec-

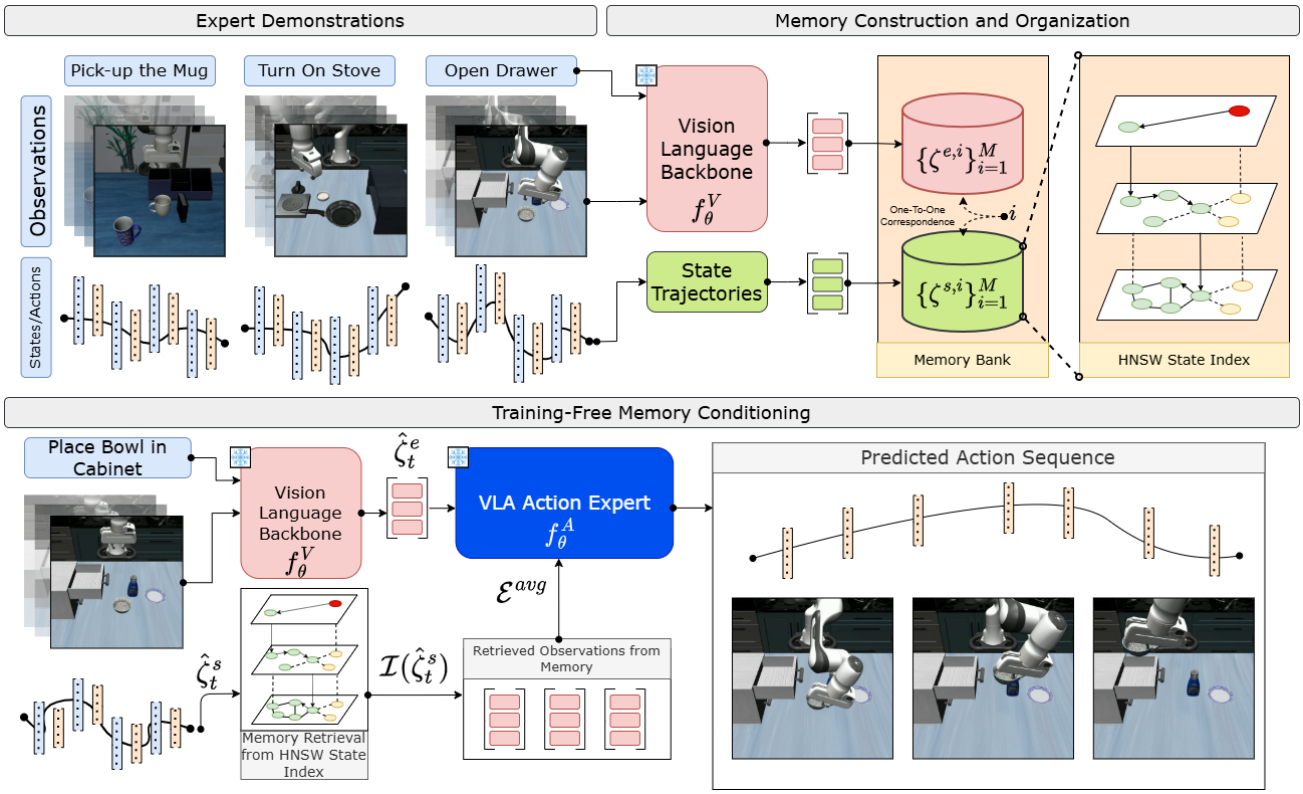



*Figure 3.* The framework of our proposed Memory. First, the expert (training) demonstrations $\mathcal{D}^{Tr}$ are used to construct and organize the memory structure (section 3 and algorithm 1). Then, the memory retrieval is performing during the inference phase via current state trajectory at timestep $t$, $\hat{\zeta}_t^s$. The indices $\mathcal{I}(\hat{\zeta}_t^s)$ are then used to retrieve the VLM feature embeddings and the action expert $f_\theta^A$ is conditioned upon via this. Note: the VLA is completely frozen during this stage leading to substantially fast runtime and minimal GPU requirements. We utilize a gating mechanism as discussed in section 3.2.2.

tion 3.2.3) only during failure to minimize inference overhead.

3. **SOTA Performance:** We demonstrate significant improvements over the no-memory baseline, achieving up to 13% absolute (27% relative) gains on standard benchmarks (section 4) for task completion success rate. Furthermore, we extend the CALVIN benchmark to task horizons of 6+ with up to 30% relative gains (section 4.3). Additional experiments on robustness of the memory to Out Of Distribution (OOD) shifts in the observations demonstrate that our method further extends the average absolute gain to 10.7%. Lastly, real world experiments on Cobot-Magic-AgileX robot demonstate up to 2X improvements over base policy.

## 2. Related Work

### 2.1. Vision Language Action Models

Recent advancements in foundational models have demonstrated that large-scale pre-training can significantly enhance robotic generalization and manipulation capabilities. Vision-Language-Action (VLA) models, such as $\pi_0$ (Black et al., 2024), RT-1 (Brohan et al., 2023b), and RT-2 (Brohan et al., 2023a), thus emerged as a natural extension from the success of standard multi-modal models. Building on this foundation, recent efforts have sought to further unify modalities for better performance; for instance, *LoHoVLA* (Yang et al., 2025b) and *Unified VLA* (Wang et al., 2025a) propose unified architectures that interleave action generation with future state prediction and planning within a single backbone. Other works like *SmolVLA* (Shukor et al., 2025) and *FlowerVLA* (Reuss et al., 2025b) have proposed lightweight architectures that optimize inference efficiency without compromising control precision. However, all these methods are subject to the aforementioned issues rendering the inference process brittle on long horizon tasks.

### 2.2. Memory-Augmented Frameworks

Methods integrating memory have thus emerged as a critical research direction. *MAP-VLA* (Li et al., 2025c) demonstrates that retrieving relevant demonstration segments can effectively guide the generation process, albeit by training hyper task specific prompts via gradient updates. *MemoryVLA* (Shi et al., 2025), *EchoVLA* (Lin et al.,

2025), and *HAMLET* (Koo et al., 2025) introduce dedicated history-aware architectures—ranging from read-write memory banks to moment tokens—that enable agents to maintain temporal consistency over extended episodes. Concurrently, retrieval mechanisms have been employed to generalize skills; (Li et al., 2025a) and its extensions, (Kuang et al., 2024) utilize hierarchical retrieval to transfer 2D affordances to 3D actions, while *ARRC* (Vorobiov et al., 2025) and EmbodiedRAG (Xie et al., 2025) leverage Retrieval-Augmented Generation (RAG) to ground language instructions in semantic maps for high-level planning.

While promising, these approaches rely on gradient updates or test-time adaptation to integrate the retrieved knowledge. For example, *STRAP* (Memmel et al., 2025) retrieves sub-trajectories but requires training a model on relevant data at deployment time, and *ExpReS-VLA* (Syed et al., 2025) utilizes experience replay buffers to fine-tune the model to prevent catastrophic forgetting. DAAAM (Gorlo et al., 2025) focuses heavily on constructing complex semantic 4D scene graphs operating in different environments than generalized robotics tasks, highlighting a different paradigm.

## 3. Methodology

### 3.1. Preliminaries

We model robotic manipulation as a (partially observable) sequential decision-making problem, where a VLA policy $f_\theta$ maps multimodal observations $o_t$ (vision and language) to actions $a_t$ executed in the environment. At each timestep $t$, the VLA receives an observation $o_t = [I_t, l_t, s_t]$ comprising of the RGB image $I_t$ from the perception mechanism, the language instruction $l_t$ and the robot's state $s_t$, where $s_t \in \mathbb{R}^{d_s}$ (discussed further in appendix section C). For many general purpose robots, the RGB images are captured from 2 angles, an overview image $I_t^1$ and a wrist image $I_t^2$. We collect both images as $I_t = (I_t^1, I_t^2)$. The goal is to predict the future action $a_t$. Generally, to accelerate the training process and make it stable, the action head predicts a chunk of future actions (Black et al., 2024). We denote the VLM backbone of the VLA as $f_\theta^V$ and correspondingly the action prediction components as $f_\theta^A$. Additionally we represent the compressed embeddings for the observations $o_t$ as $e_t = f_\theta^V(o_t), e_t \in \mathbb{R}^{d_e}$ over which the action head operates $a_t = f_\theta^A(e_t)$. Since our method operates purely at inference time and treats the VLA as a frozen model, thus we also omit the discussion about the training objectives of the underlying models and refer the reader to (Kawaharazuka et al., 2025) for a detailed review.

As discussed previously that the first order Markovian property results in severe lack of temporal consistency in these models, we will consider sequential timestep chunks of length $\bar{H}$ in the entire memory algorithm discussed below. Thus, to extend the notations, we denote $\zeta_t^e = (e_{t-\bar{H}+1}, ...e_t)$. Similarly we denote $\zeta_t^s = (s_{t-\bar{H}+1}, ...s_t)$.

### 3.2. Proposed Solution: Training-Free Memory

As previously discussed, in order to avoid computational cost of fine-tuning and keeping the memory framework agnostic to any specific VLA, we focus on a **hybrid, training-free** and *external* memory. We start by discussing the memory construction and organization phase (section 3.2.1) which leverages $\zeta_t^e$ as the compressed summary in conjunction with the robot's state (consisting of coordinates and meta information). Since $\mathcal{M}_{ext}$ can expand rapidly, we need an effective retrieval mechanism that operates well in high dimensions. To this end, we leverage Hierarchical Navigable Small World (HNSW) as the data structure with $L_2$ as the metric of choice. Subsequently, we discuss the retrieval mechanism (section 3.2.2) which leverages the robot states $\zeta_t^s$ during the inference rollout to retrieve the most similar situations encountered by robot in the past. HNSW plays a crucial role due to the high recall property in this case. Lastly, we discuss the action generation via $f_\theta^A$ conditioned on the retrieved embeddings $\zeta_t^e$ (section 3.2.3) and an appropriate fallback mechanism that provides a calibrated runtime-success tradeoff. The memory architecture is detailed in figure 3.

#### 3.2.1. Memory Construction and Organization

We consider a collection of expert (training) trajectories $\mathcal{D}^{Tr} = \{\tau_1, \ldots, \tau_N\}$, where each trajectory $\tau_i = (o_1^i, \ldots, o_{L(\tau_i)}^i)$ is a sequence of observations with trajectory-specific length $L(\tau_i)$. Following standard practice (Reuss et al., 2025b; Black et al., 2024; Marafioti et al., 2025), we partition each trajectory $\tau_i$ into overlapping sub-trajectories using a sliding window of fixed horizon $\bar{H}$ and stride 1. Concretely, this produces $L(\tau_i) - \bar{H} + 1$ sub-trajectories of the form $(o_{t-\bar{H}+1}^i, \ldots, o_t^i)$. We treat each sub-trajectory as an separate instance in the memory with no dependence. Furthermore, each sub-trajectory is encoded into a pair of embeddings: a VLM embedding $\zeta^e$ and a corresponding state embedding $\zeta^s$ as defined previously. Consolidating all trajectories and windows yields a simplified version of the memory

$$\mathcal{M}_{ext}^{raw} = \{(\zeta^{e,1}, \zeta^{s,1}), \ldots, (\zeta^{e,M}, \zeta^{s,M})\} \quad (1)$$

where $M = |\mathcal{M}_{ext}| = |\mathcal{M}_{ext}^{raw}| = \sum_{i=1}^{N}(L(\tau_i) - \bar{H} + 1)$ is fixed after processing the training dataset. Variation in $M$ is empirically discussed in section D.2.2.

At inference time, we denote by $\hat{\tau}_t = (\hat{o}_1, \ldots, \hat{o}_t)$ the partially unrolled trajectory up to timestep $t$ in the environment. The corresponding state embedding $\hat{\zeta}_t^s$, computed over the most recent $\bar{H}$ observations (or fewer at early timesteps), is used as the query for memory retrieval.

To construct $\mathcal{M}_{ext}$ for efficient retrieval, we index the em-

bedding pairs in $\mathcal{M}_{ext}^{raw}$. As discussed earlier in the context of non-robust VLM backbones, the visual embeddings $e$ (and hence $\zeta^e$) are insufficiently discriminative for fine-grained retrieval and are therefore not used as retrieval keys (see section H.2 for further discussion). Instead, retrieval is conditioned solely on the state embeddings $\zeta^s$, with the associated $\zeta^e$ retrieved as values. Such proprioceptive state inputs have also been studied recently (Yu et al., 2026; Kim et al., 2025; Li et al., 2026), justifying its practicality.

Since $\mathcal{M}_{ext}^{raw}$ can be large and retrieval must be performed at low latency during inference, we employ approximate nearest neighbor (ANN) search over the set of state embeddings $\{\zeta^{s,1}, \ldots, \zeta^{s,M}\}$. We use Hierarchical Navigable Small World (HNSW) indexing (Malkov & Yashunin, 2020), which offers favorable scaling and high recall for high-dimensional ANN search using the $L_2$ distance metric. We adopt the HNSW implementation provided by FAISS (Johnson et al., 2021).

#### 3.2.2. Memory Retrieval

We first retrieve the indices of the top-$k$ neighbors via $\hat{\zeta}_t^s$ of $\hat{\tau}_t$ using the $\mathcal{O}(\log(M))$ search in HNSW. Calling this set of indices as $\mathcal{I}(\hat{\zeta}_t^s)$, we then retrieve the corresponding embeddings $\mathcal{E}(\hat{\zeta}_t^s, k) = \{\zeta^{e,i}\}, i \in \mathcal{I}(\hat{\zeta}_t^s)$. Formally, we have

$$\mathcal{I}(\hat{\zeta}_t^s) = \arg\text{min-k}\{\mathbb{D}(\hat{\zeta}_t^s, \zeta^{s,i})\}_{i=1}^{M} \tag{2}$$

While computing the distance using $\mathbb{D}(\cdot, \cdot)$, we simply flatten the temporal dimension of $\hat{\zeta}_t^s$ and $\zeta^s$ which is causally consistent due to use of running window of previous $\bar{H}$ timesteps during construction.

#### 3.2.3. Memory Conditioned Action Generation

Post retrieval, we have multiple design choices for incorporating $\mathcal{E}(\hat{\zeta}_t^s, k)$ in the action generation module $f_\theta^A$. However, to realize a practical conditioning method, we first average the $k$ embeddings in $\mathcal{E}(\hat{\zeta}_t^s, k)$ to obtain $\mathcal{E}^{avg}(\hat{\zeta}_t^s, k) \in \mathbb{R}^{\bar{H} \times d_e}$ and then concatenate to $\hat{e}_t$ (embedding at timestep $t$ for inference trajectory $\hat{\tau}$) at the temporal dimension, thus obtaining the action prediction (or action chunk prediction) via $\hat{a}_t^{mem} = f_\theta^A(\mathcal{E}^{avg} || \hat{e}_t)$. The concatenation remains valid as the embedding dimensions $d_e$ are consistent across the entire pipeline. The advantage of this conditioning method naturally accounts for the complex interplay of the attention mechanism used in the action prediction heads as it then implicitly attends to the most relevant components of $\mathcal{E}^{avg}$.

We succinctly synthesize the pipeline in Algorithms 1in appendix section D.

#### 3.2.4. Complexity Analysis

**Space Complexity:** Using the notations above, the total space complexity is $\mathcal{O}\big(M\,\bar{H}(d_s + d_e) + M \log M\big)$ accounting for $\zeta^s \in \mathbb{R}^{\bar{H} \times d_s}$ and $\zeta^e \in \mathbb{R}^{\bar{H} \times d_e}$ and the second term considers the HNSW index (ignoring the max degree as we keep it a constant).

**Time Complexity:** For a single query with $\hat{\zeta}_t^s \in \mathbb{R}^{\bar{H} \times d_s}$, the retrieval time complexity is $\mathcal{O}\big((\log M)\,\bar{H} d_s + k\,\bar{H} d_e\big)$ corresponding to ANN search and averaging of the retrieved embeddings to obtain $\mathcal{E}^{avg}(\hat{\zeta}_t^s, k)$. These operations incure trivial costs when implemented in well optimized low level libraries such as FAISS.

#### 3.2.5. Memory Conditioning

Retrieving $\mathcal{E}(\hat{\zeta}_t^s, k)$ and conditioning $f_\theta^A$ on its average $\mathcal{E}^{avg}(\hat{\zeta}_t^s, k)$ provides rich context; however, it also invokes memory for simpler tasks solvable without external assistance. We therefore employ a selective mechanism that triggers memory conditioning only upon base policy failure. Let the base action be $\hat{a}_t = f_\theta^A(\hat{e}_t)$, and the memory-augmented action be $\hat{a}_t^{mem}$. We define the output action $\hat{a}_t^{out}$ as:

$$\hat{a}_t^{out} = \begin{cases} \hat{a}_t, & \text{if } TaskState(\hat{\tau}_t) = 1, \\ \hat{a}_t^{mem}, & \text{if } TaskState(\hat{\tau}_t) = 0, \end{cases} \tag{3}$$

where $TaskState(\hat{\tau}_t)$ is a binary classifier utilized to realize the completion of the task over $\hat{\tau}_t$, characterizing whether the goal conditions were satisfied (see Section D for details). As shown in Section 4.2, this fallback strategy incurs negligible overhead in completion time (see Figures 17 and 16). We further quantify retrieval requests in Section D.2 and provide additional characterizations in Section H.

### 3.3. Advantages of this Framework

Typically, any algorithm that improves downstream performance post fine-tuning will essentially require either *access to external information* or *adjustment of the model weights* in some form. However, the latter induces several complexities around catastrophic forgetting, unlearning, and high sensitivity to data and tasks (Kotha et al., 2024; Zhang et al., 2024; Liu et al., 2024). The last challenge becomes a bottleneck in robotics applications. Due to these reasons, the former setup leveraged in our work, has garnered huge attention and is the premise behind the entire domain of RAG and Episodic Memory now mainstream in industrial deployments. Additionally, figure 9 clearly demonstrates that even a *small amount* of fine-tuning data used to construct the memory leads to notable gains.

*Table 1.* Results across the CALVIN suite for all the three datasets. *"w/o"* represents numbers without memory whereas *"w"* represents conditioning on memory. Std deviation is provided in the parentheses and the mean values represent the *Success rate* numbers in %. "Avg. Len." is computed over number of instructions successfully completed per sequence (out of 5). Additional baselines in table 13.

| Train→Test | Method | No. Instructions in a Row | | | | | Avg. Len. |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | |
| ABC→D | MoDE w/o Mem | 96.6 (0.278) | 89.5 (0.928) | 81.9 (0.485) | 73.5 (0.565) | 64.3 (0.307) | 4.06 (1.49) |
| | MoDE w Mem | **96.8** (0.438) | **92.4** (0.931) | **86.2** (0.676) | **79.7** (0.726) | **71.6** (0.309) | **4.27** (1.37) |
| | FLOWER w/o Mem | 99.2 (0.304) | 96.2 (0.826) | 90.1 (0.541) | 83.9 (0.587) | 77.6 (0.237) | 4.47 (1.12) |
| | FLOWER w Mem | **99.8** (0.294) | **97.3** (0.834) | **93.3** (0.516) | **88.1** (0.561) | **82.3** (0.295) | **4.61** (0.96) |
| ABCD→D | MoDE w/o Mem | 97.4 (0.346) | 92.7 (0.898) | 89.3 (0.512) | 86.1 (0.700) | 81.3 (0.391) | 4.47 (1.27) |
| | MoDE w Mem | **97.6** (0.417) | **95.3** (0.879) | **93.2** (0.512) | **91.4** (0.621) | **88.5** (0.283) | **4.66** (1.07) |
| | FLOWER w/o Mem | 98.5 (0.310) | 95.6 (0.773) | 92.4 (0.463) | 89.4 (0.558) | 85.3 (0.215) | 4.61 (1.07) |
| | FLOWER w Mem | **99.9** (0.343) | **98.5** (0.784) | **96.7** (0.523) | **94.5** (0.596) | **91.8** (0.239) | **4.81** (0.70) |
| D→D | MoDE w/o Mem | 90.9 (0.274) | 77.6 (0.813) | 65.5 (0.602) | 56.0 (0.533) | 48.6 (0.249) | 3.39 (1.83) |
| | MoDE w Mem | **94.3** (0.285) | **85.6** (0.893) | **78.0** (0.678) | **70.6** (0.729) | **61.9** (0.382) | **3.90** (1.64) |
| | FLOWER w/o Mem | 98.4 (0.356) | 93.3 (0.819) | 86.6 (0.534) | 80.0 (0.543) | 72.3 (0.217) | 4.31 (1.30) |
| | FLOWER w Mem | **99.6** (0.347) | **95.6** (0.806) | **90.9** (0.535) | **86.1** (0.594) | **80.2** (0.285) | **4.52** (1.09) |

*Table 2.* Results over LIBERO-long and LIBERO-90 respectively. *"w/o"* represents numbers without memory whereas *"w"* represents conditioning on memory. Mean Success rate numbers are represented in % (with std dev). Additional baselines is reported in table 12.

| Dataset | OpenVLA | CogACT | $\pi_0$ | MemoryVLA | MAP-VLA | MoDE w/o Mem | MoDE w Mem | Flower w/o Mem | Flower w Mem |
|---|---|---|---|---|---|---|---|---|---|
| LIBERO-long | 56.21 | 87.42 | 86.33 | 93.4 | 83.4 (0.70) | 90.81 (0.73) | **96.00** (0.39) | 91.00 (1.06) | **96.80** (0.41) |
| LIBERO-90 | 74.19 | 92.16 | 92.41 | 95.6 | - | 94.28 (0.64) | **97.28** (0.44) | 94.12 (0.70) | **97.17** (0.59) |

# 4. Experiments

## 4.1. Setup

We experiment on multiple challenging benchmarks from the CALVIN (Mees et al., 2022) and LIBERO (Liu et al., 2023) suite. We select two models: FlowerVLA (Reuss et al., 2025b), a lightweight and efficient VLA policy with state-of-the-art results using fewer than 1B parameters, and MoDE (Reuss et al., 2025a), which leverages a mixture of experts in the action prediction component. Both environments utilize a Franka Panda robot with 7-DOF in the simulator, which aligns with a substantial fraction of real-world robotics tasks and is widely used in the literature (Shi et al., 2025; Li et al., 2025c; Reuss et al., 2025b). We retain the standard evaluation protocol from prior work, reporting success rates across tasks (Reuss et al., 2025b; Shi et al., 2025), along with the mean and standard deviation of the average sequence length on CALVIN datasets (Reuss et al., 2025b). We generate 1000 evaluation sequences for CALVIN following (Reuss et al., 2025b), and for LIBERO we report success rates over all tasks using 50 rollouts per task (Black et al., 2024; Reuss et al., 2025b; Li et al., 2025c). All experiments are conducted with three different random seeds, and results are globally aggregated as mean and standard deviation.

## 4.2. Main Results

Table 1 presents the results across the *Number of Instructions in a Row* for CALVIN across the splits *D→D*, *ABC→D* and *ABCD→D* datasets. The results for LIBERO-long and LIBERO-90 long horizon datasets are provided in table 2. Additional results over popular VLAs from the literature - OpenVLA (Kim et al., 2024), $\pi_0$ (Black et al., 2024), CogACT (Li et al., 2024) are also provided for completeness of the results and provided simply for benchmarking purposes. For each dataset, we only use the corresponding training set demonstrations to construct $\mathcal{M}_{ext}$. For eg - CALVIN *ABC→D* memory construction depends solely on the *ABC* training data, and is *agnostic* of the memory construction and evaluation of other split. This serves as a proxy experiment for an OOD setup and task variation where the memory conditioning adapts the model for improved downstream performance. Same procedure is used for LIBERO. Note that MoDE and FlowerVLA perform better than these models even in *no memory* setting, thus *justifying* our selection of the latter for evaluation. Lastly, even though recent memory enhanced VLAs are **not aligned** with our training-free and model-algorithm agnostic setup, we still provide the results over (Shi et al., 2025; Li et al., 2025c) in table 2 for completeness. Further comparisons are also discussed in tables 13 and 12 in the appendix.

**Success Rates:** We observe a clear trend that increasing task complexity (measured by the number of chained instructions) amplifies the benefits of memory conditioning. On CALVIN, the largest absolute gains are observed on the challenging *D→D* split—where training data is limited to a single split—with improvements of 13.3% for 5 instructions and 14.6% for 4 instructions over MoDE, while Flower shows consistent gains of up to 7.9%. When conditioned on memory, single-instruction performance nearly saturates, achieving close to complete success, as seen for Flower on *ABCD→D*. LIBERO benchmarks exhibit a similar pat-

*Table 3.* Extension of the CALVIN benchmark to longer task horizons (6-10 instructions) on a separate inference set. The results follow the same notation as table 1.

| Train→Test | Method | No. Instructions in a Row | | | | | Avg. Len. |
|---|---|---|---|---|---|---|---|
| | | 6 | 7 | 8 | 9 | 10 | |
| ABC→D | MoDE w/o Mem | 58.8 (0.45) | 55.6 (0.88) | 52.8 (0.61) | 50.0 (0.56) | 44.8 (0.23) | 6.74 (3.51) |
| | MoDE w Mem | **69.2** (0.47) | **63.0** (0.92) | **59.9** (0.63) | **53.7** (0.62) | **50.2** (0.34) | **7.16** (3.37) |
| | FLOWER w/o Mem | 80.8 (0.44) | 76.4 (1.18) | 70.0 (0.77) | 67.8 (0.84) | 61.2 (0.34) | 8.18 (2.76) |
| | FLOWER w Mem | **86.4** (0.44) | **84.0** (1.28) | **76.8** (0.75) | **73.6** (0.79) | **67.2** (0.46) | **8.64** (2.45) |
| ABCD→D | MoDE w/o Mem | 75.2 (0.36) | 74.0 (0.79) | 71.2 (0.47) | 66.8 (0.59) | 61.2 (0.22) | 7.94 (3.16) |
| | MoDE w Mem | **89.6** (0.47) | **83.2** (0.86) | **78.8** (0.59) | **74.8** (0.67) | **70.8** (0.27) | **8.72** (2.45) |
| | FLOWER w/o Mem | 89.6 (0.42) | 86.8 (0.97) | 84.4 (0.55) | 81.2 (0.81) | 77.6 (0.32) | 8.93 (2.46) |
| | FLOWER w Mem | **96.8** (0.37) | **95.6** (1.10) | **92.8** (0.62) | **90.4** (0.79) | **88.0** (0.38) | **9.60** (1.27) |
| D→D | MoDE w/o Mem | 46.4 (0.35) | 43.6 (0.95) | 41.2 (0.60) | 34.4 (0.59) | 29.2 (0.39) | 5.51 (3.76) |
| | MoDE w Mem | **58.4** (0.49) | **50.8** (0.93) | **42.4** (0.53) | **36.0** (0.58) | **31.6** (0.31) | **6.19** (3.46) |
| | FLOWER w/o Mem | 62.8 (0.55) | 60.8 (1.13) | 58.0 (0.73) | 54.4 (1.02) | 51.6 (0.56) | 7.14 (3.38) |
| | FLOWER w Mem | **82.0** (0.52) | **80.4** (1.53) | **76.4** (0.86) | **71.6** (1.13) | **66.4** (0.57) | **8.40** (2.74) |

*Table 4.* Results for the Real World Evaluation. We report the success and improvements in a step-wise manner similar to CALVIN, wherein failure at a sub-task results in the remaining tasks being considered unsuccessful.

| | Reach Microwave Door | | Open Microwave Door | | Place Bowl Inside | |
|---|---|---|---|---|---|---|
| | No Memory | With Memory | No Memory | With Memory | No Memory | With Memory |
| Success Rate (in %) | 80% | **90%** | 30% | **60%** | 20% | **30%** |
| Absolute Gains | - | **10%** | - | **30%** | - | **10%** |
| Relative Gains | - | **12.5%** | - | **100%** | - | **50%** |

tern: despite strong baselines above 90% for both MoDE and Flower, memory conditioning yields additional gains of up to 5.8%. These non-trivial improvements are very significant in real-world deployments.

**Timesteps taken:** We evaluate efficiency using the number of timesteps required for successful task completion, capturing the runtime–success tradeoff. Across CALVIN and LIBERO (figs 16 and 17), conditioning on memory incurs negligible overhead, restricted to under 2% on CALVIN and 3.5% on LIBERO in the median timesteps. Coupled with the significant success-rate gains and only a marginal increase in GPU memory usage, these results demonstrate the practical viability of the proposed memory framework.

### 4.3. CALVIN Extension to 6+ Instructions

As an additional contribution, we extend the CALVIN benchmark to task horizon length of 6 to 10 to demonstrate that the gains from memory conditioning strengthen the performance. The specific methodology and example instructions chains are provided in supplementary section G while the results are shown in table 3. These experiments are conducted on a separate set of 250 sequences to decouple from results in table 1 and perform a broader validation. We note consistent gains with up to 30% relative improvement on D→D split over Flower VLA, which remain elevated till 10 instruction chains. MoDE exhibits a similar pattern for length 6 and 7 while exhibiting reduced margins which we attribute to the weakness of VLA model itself, which has low default (no-memory) results. These numbers also indicate the need to further research on longer horizons beyond the consideration of current benchmarks. Note, we used the same memory constructed corresponding to each dataset in table 1 and thus only the inference pipeline was implemented for this experiment, eliminating any overhead.

### 4.4. Real-World Experiments

We conduct real-world experiments on a *Cobot-Magic-AgileX* robot equipped with a 6-DOF arm using a first-person RGB camera setup (details in appendix B). The results are provided in table 4 where we observe up to 2X gains. Noteworthy is that we incorporate various complex OOD evaluations, 80% of the eval, in this setup as discussed further in section B.3.

## 5. Conclusion and Future Work

In this work, we propose a training-free, model-agnostic memory to augment robotic models at inference time for long horizon tasks. By performing extensive experiments across a diverse suite of challenging long horizon benchmarks, we highlight the performance improvements of up to 30% in relative terms. Our proposed framework retains improvement margins under extended task horizons and inference time distribution shifts. An important future direction is to expand into a dynamic feedback oriented memory, which is updated during inference time.

## LLM Usage

We primarily used proprietary LLMs to improve the paper writeup and final draft. Parts of the code were written using Claude and verified manually.

## References


Baddeley, A. Working memory: An overview. *Working memory and education*, pp. 1–31, 2006.

Barnett, S., Brannelly, Z., Kurniawan, S., and Wong, S. Fine-tuning or fine-failing? debunking performance myths in large language models, 2024. URL https://arxiv.org/abs/2406.11201.

Berglund, L., Tong, M., Kaufmann, M., Balesni, M., Stickland, A. C., Korbak, T., and Evans, O. The reversal curse: LLMs trained on "a is b" fail to learn "b is a". In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=GPKTIktA0k.

Black, K., Brown, N., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Groom, L., Hausman, K., Ichter, B., Jakubczak, S., Jones, T., Ke, L., Levine, S., Li-Bell, A., Mothukuri, M., Nair, S., Pertsch, K., Shi, L. X., Tanner, J., Vuong, Q., Walling, A., Wang, H., and Zhilinsky, U. $\pi_0$: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., Florence, P., Fu, C., Arenas, M. G., Gopalakrishnan, K., Han, K., Hausman, K., Herzog, A., Hsu, J., Ichter, B., Irpan, A., Joshi, N., Julian, R., Kalashnikov, D., Kuang, Y., Leal, I., Lee, L., Lee, T.-W. E., Levine, S., Lu, Y., Michalewski, H., Mordatch, I., Pertsch, K., Rao, K., Reymann, K., Ryoo, M., Salazar, G., Sanketi, P., Sermanet, P., Singh, J., Singh, A., Soricut, R., Tran, H., Vanhoucke, V., Vuong, Q., Wahid, A., Welker, S., Wohlhart, P., Wu, J., Xia, F., Xiao, T., Xu, P., Xu, S., Yu, T., and Zitkovich, B. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023a. URL https://arxiv.org/abs/2307.15818.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Dabis, J., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jackson, T., Jesmonth, S., Joshi, N. J., Julian, R., Kalashnikov, D., Kuang, Y., Leal, I., Lee, K.-H., Levine, S., Lu, Y., Malla, U., Manjunath, D., Mordatch, I., Nachum, O., Parada, C., Peralta, J., Perez, E., Pertsch, K., Quiambao, J., Rao, K., Ryoo, M., Salazar, G., Sanketi, P., Sayed, K., Singh, J., Sontakke, S., Stone, A., Tan, C., Tran, H., Vanhoucke, V., Vega, S., Vuong, Q., Xia, F., Xiao, T., Xu, P., Xu, S., Yu, T., and Zitkovich, B. Rt-1: Robotics transformer for real-world control at scale, 2023b. URL https://arxiv.org/abs/2212.06817.

Chen, J., Song, W., Ding, P., Zhou, Z., Zhao, H., Tang, F., Wang, D., and Li, H. Unified diffusion vla: Vision-language-action model via joint discrete denoising diffusion process, 2025. URL https://arxiv.org/abs/2511.01718.

Cherepanov, E., Kachaev, N., Kovalev, A. K., and Panov, A. I. Memory, benchmark robots: A benchmark for solving complex tasks with reinforcement learning, 2025. URL https://arxiv.org/abs/2502.10550.

Fu, M., Huang, H., Datta, G., Chen, L. Y., Panitch, W., Liu, F., Li, H., and Goldberg, K. Icrt: In-context imitation learning via next-token prediction. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5937–5944, 2025. doi: 10.1109/ICRA55743.2025.11128272.

Gorlo, N., Schmid, L., and Carlone, L. Describe anything anywhere at any moment, 2025. URL https://arxiv.org/abs/2512.00565.

Hancock, A. J., Ren, A. Z., and Majumdar, A. Run-time observation interventions make vision-language-action models more visually robust. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9499–9506. IEEE, 2025.

Jiang, P., Ergu, D., Liu, F., Cai, Y., and Ma, B. A review of yolo algorithm developments. *Procedia Computer Science*, 199:1066–1073, 2022. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2022.01.135. URL https://www.sciencedirect.com/science/article/pii/S1877050922001363. The 8th International Conference on Information Technology and Quantitative Management (ITQM 2020 2021): Developing Global Digital Economy after COVID-19.

Johnson, J., Douze, M., and Jégou, H. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2021. doi: 10.1109/TBDATA.2019.2921572.

Kawaharazuka, K., Oh, J., Yamada, J., Posner, I., and Zhu, Y. Vision-language-action models for robotics: A review towards real-world applications. *IEEE Access*, 13:162467–162504, 2025. doi: 10.1109/ACCESS.2025.3609980.

Kim, M. J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E., Lam, G., Sanketi, P., Vuong, Q., Kollar, T., Burchfiel, B., Tedrake,

R., Sadigh, D., Levine, S., Liang, P., and Finn, C. Openvla: An open-source vision-language-action model, 2024. URL https://arxiv.org/abs/2406.09246.

Kim, T., Lee, J., Koo, M., Kim, D., Lee, K., Kim, C., Seo, Y., and Shin, J. Contrastive representation regularization for vision-language-action models, 2025. URL https://arxiv.org/abs/2510.01711.

Koo, M., Choi, D., Kim, T., Lee, K., Kim, C., Seo, Y., and Shin, J. Hamlet: Switch your vision-language-action model into a history-aware policy, 2025. URL https://arxiv.org/abs/2510.00695.

Kotha, S., Springer, J. M., and Raghunathan, A. Understanding catastrophic forgetting in language models via implicit inference. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VrHiF2hsrm.

Kuang, Y., Ye, J., Geng, H., Mao, J., Deng, C., Guibas, L., Wang, H., and Wang, Y. Ram: Retrieval-based affordance transfer for generalizable zero-shot robotic manipulation, 2024. URL https://arxiv.org/abs/2407.04689.

Leon, M. Gpt-5 and open-weight large language models: Advances in reasoning, transparency, and control. *Information Systems*, 136:102620, 2026. ISSN 0306-4379. doi: https://doi.org/10.1016/j.is.2025.102620. URL https://www.sciencedirect.com/science/article/pii/S0306437925001061.

Li, D., Zhang, Y., Cao, M., Liu, D., Xie, W., Hui, T., Lin, L., Xie, Z., and Li, Y. Towards long-horizon vision-language-action system: Reasoning, acting and memory. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6839–6848, 2025a.

Li, H., Yang, S., Chen, Y., Chen, X., Yang, X., Tian, Y., Wang, H., Wang, T., Lin, D., Zhao, F., and Pang, J. Cronusvla: Towards efficient and robust manipulation via multi-frame vision-language-action modeling, 2025b. URL https://arxiv.org/abs/2506.19816.

Li, Q., Liang, Y., Wang, Z., Luo, L., Chen, X., Liao, M., Wei, F., Deng, Y., Xu, S., Zhang, Y., Wang, X., Liu, B., Fu, J., Bao, J., Chen, D., Shi, Y., Yang, J., and Guo, B. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation, 2024. URL https://arxiv.org/abs/2411.19650.

Li, R., Guo, W., Wu, Z., Wang, C., Deng, H., Weng, Z., Tan, Y.-P., and Wang, Z. Map-vla: Memory-augmented prompting for vision-language-action model in robotic manipulation, 2025c. URL https://arxiv.org/abs/2511.09516.

Li, Z., Li, Y., Jiang, J.-J., Zhou, L., Zhang, T., and Zheng, W.-S. Revip: Reducing false completion in vision-language-action models with vision-proprioception rebalance, 2026. URL https://arxiv.org/abs/2601.16667.

Lin, M., Liang, X., Lin, B., Jingzhi, L., Jiao, Z., Li, K., Ma, Y., Liu, Y., Zhao, S., Zhuang, Y., and Liang, X. Echovla: Robotic vision-language-action model with synergistic declarative memory for mobile manipulation, 2025. URL https://arxiv.org/abs/2511.18112.

Liu, B., Zhu, Y., Gao, C., Feng, Y., Liu, Q., Zhu, Y., and Stone, P. Libero: Benchmarking knowledge transfer for lifelong robot learning. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 44776–44791. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/8c3c666820ea055a77726d66fc7d447f-Paper-Datasets_and_Benchmarks.pdf.

Liu, C., Zhang, J., Li, C., Zhou, Z., Wu, S., Huang, S., and Duan, H. Ttf-vla: Temporal token fusion via pixel-attention integration for vision-language-action models, 2025. URL https://arxiv.org/abs/2508.19257.

Liu, S., Yao, Y., Jia, J., Casper, S., Baracaldo, N., Hase, P., Yao, Y., Liu, C. Y., Xu, X., Li, H., Varshney, K. R., Bansal, M., Koyejo, S., and Liu, Y. Rethinking machine unlearning for large language models, 2024. URL https://arxiv.org/abs/2402.08787.

Luo, F. Vision-language models for robot success detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(21):23750–23752, Mar. 2024. doi: 10.1609/aaai.v38i21.30552. URL https://ojs.aaai.org/index.php/AAAI/article/view/30552.

Ma, Y., Song, Z., Zhuang, Y., Hao, J., and King, I. A survey on vision-language-action models for embodied ai, 2026. URL https://arxiv.org/abs/2405.14093.

Mahajan, S., Kadambi, S., Le, H., Hayat, M., and Porikli, F. Do-undo: Generating and reversing physical actions in vision-language models, 2025. URL https://arxiv.org/abs/2512.13609.

Malkov, Y. A. and Yashunin, D. A. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836, 2020. doi: 10.1109/TPAMI.2018.2889473.

Marafioti, A., Zohar, O., Farré, M., Noyan, M., Bakouch, E., Cuenca, P., Zakka, C., Allal, L. B., Lozhkov, A., Tazi, N., Srivastav, V., Lochner, J., Larcher, H., Morlon, M., Tunstall, L., von Werra, L., and Wolf, T. Smolvlm: Redefining small and efficient multimodal models, 2025. URL https://arxiv.org/abs/2504.05299.

Mees, O., Hermann, L., Rosete-Beas, E., and Burgard, W. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022. doi: 10.1109/LRA.2022.3180108.

Memmel, M., Berg, J., Chen, B., Gupta, A., and Francis, J. Strap: Robot sub-trajectory retrieval for augmented policy learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Nayak, S., Orozco, A. M., Have, M. T., Thirumalai, V., Zhang, J., Chen, D., Kapoor, A., Robinson, E., Gopalakrishnan, K., Harrison, J., Ichter, B., Mahajan, A., and Balakrishnan, H. Llamar: Long-horizon planning for multi-agent robots in partially observable environments, 2025. URL https://arxiv.org/abs/2407.10031.

Parthasarathy, V. B., Zafar, A., Khan, A., and Shahid, A. The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities, 2024. URL https://arxiv.org/abs/2408.13296.

Rahmanzadehgervi, P., Bolton, L., Taesiri, M. R., and Nguyen, A. T. Vision language models are blind. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, pp. 18–34, December 2024.

Reuss, M., Pari, J., Agrawal, P., and Lioutikov, R. Efficient diffusion transformer policies with mixture of expert denoisers for multitask learning. In Yue, Y., Garg, A., Peng, N., Sha, F., and Yu, R. (eds.), *International Conference on Learning Representations*, volume 2025, pp. 17247–17275, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/2bfc4acb2e5c4af1576d14b0df9a664c-Paper-Conference.pdf.

Reuss, M., Zhou, H., Rühle, M., Ömer Erdinç Yağmurlu, Otto, F., and Lioutikov, R. Flower: Democratizing generalist robot policies with efficient vision-language-action flow policies, 2025b. URL https://arxiv.org/abs/2509.04996.

Reyna, V. and Brainerd, C. Fuzzy-trace theory: An interim synthesis. *Learning and Individual Differences*, 7(1):1–75, 1995. ISSN 1041-6080. doi: https://doi.org/10.1016/1041-6080(95)90031-4. URL https://www.sciencedirect.com/science/article/pii/1041608095900314. Special Issue: Fuzzy-Trace Theory.

Sapkota, R., Cao, Y., Roumeliotis, K. I., and Karkee, M. Vision-language-action models: Concepts, progress, applications and challenges, 2025. URL https://arxiv.org/abs/2505.04769.

Scalise, R., Thomason, J., Bisk, Y., and Srinivasa, S. S. Improving robot success detection using static object data. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2019, Macau, SAR, China, November 3-8, 2019*, pp. 4229–4235. IEEE, 2019. doi: 10.1109/IROS40897.2019.8968142. URL https://doi.org/10.1109/IROS40897.2019.8968142.

Shafiei, M., Saffari, H., and Moosavi, N. S. More or less wrong: A benchmark for directional bias in llm comparative reasoning, 2025. URL https://arxiv.org/abs/2506.03923.

Shi, H., Xie, B., Liu, Y., Sun, L., Liu, F., Wang, T., Zhou, E., Fan, H., Zhang, X., and Huang, G. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation, 2025. URL https://arxiv.org/abs/2508.19236.

Shi, W., Min, S., Yasunaga, M., Seo, M., James, R., Lewis, M., Zettlemoyer, L., and Yih, W.-t. REPLUG: Retrieval-augmented black-box language models. In Duh, K., Gomez, H., and Bethard, S. (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8371–8384, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.463. URL https://aclanthology.org/2024.naacl-long.463/.

Shukor, M., Aubakirova, D., Capuano, F., Kooijmans, P., Palma, S., Zouitine, A., Aractingi, M., Pascal, C., Russi, M., Marafioti, A., Alibert, S., Cord, M., Wolf, T., and Cadene, R. Smolvla: A vision-language-action model for affordable and efficient robotics, 2025. URL https://arxiv.org/abs/2506.01844.

Suzuki, K., Nakamurama, H., Miyamoto, K., and Ogata, T. Compact task-aligned imitation learning for laboratory automation, 2026. URL https://arxiv.org/abs/2603.01110.

Syed, S. N., Ahuja, Y., Jakobsson, A., and Ichnowski, J. Expres-vla: Specializing vision-language-action models through experience replay and retrieval, 2025. URL https://arxiv.org/abs/2511.06202.

Vo, K., Hanyu, T., Ikebe, Y., Pham, T. T., Chung, N., Vu, M. N., Minh, D. N. H., Nguyen, A., Gunderman, A., Rainwater, C., and Le, N. Clutter-resistant vision-language-action models through object-centric and geometry grounding, 2025. URL https://arxiv.org/abs/2512.22519.

Vorobiov, E., Mahmood, A. J., Rezvani, S., and Chhabra, R. Arrc: Advanced reasoning robot control - knowledge-driven autonomous manipulation using retrieval-augmented generation, 2025. URL https://arxiv.org/abs/2510.05547.

Wang, Y., Li, X., Wang, W., Zhang, J., Li, Y., Chen, Y., Wang, X., and Zhang, Z. Unified vision-language-action model, 2025a. URL https://arxiv.org/abs/2506.19850.

Wang, Z., Zhou, Z., Song, J., Huang, Y., Shu, Z., and Ma, L. Vlatest: Testing and evaluating vision-language-action models for robotic manipulation. *Proc. ACM Softw. Eng.*, 2(FSE), June 2025b. doi: 10.1145/3729343. URL https://doi.org/10.1145/3729343.

Xie, Q., Min, S. Y., Ji, P., Yang, Y., Zhang, T., Xu, K., Bajaj, A., Salakhutdinov, R., Johnson-Roberson, M., and Bisk, Y. Embodied-rag: General non-parametric embodied memory for retrieval and generation, 2025. URL https://arxiv.org/abs/2409.18313.

Yan, S.-Q., Gu, J.-C., Zhu, Y., and Ling, Z.-H. Corrective retrieval augmented generation, 2024. URL https://arxiv.org/abs/2401.15884.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

Yang, Y., Sun, J., Kou, S., Wang, Y., and Deng, Z. Lohovla: A unified vision-language-action model for long-horizon embodied tasks, 2025b. URL https://arxiv.org/abs/2506.00411.

Yin, Y., Wang, Z., Sharma, Y., Niu, D., Darrell, T., and Herzig, R. In-context learning enables robot action prediction in llms. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8972–8979, 2025. doi: 10.1109/ICRA55743.2025.11128807.

Yu, B., Lian, S., Lin, X., Wei, Y., Shen, Z., Wu, C., Miao, Y., Wang, X., Wang, B., Huang, C., and Chen, K. Twinbrainvla: Unleashing the potential of generalist vlms for embodied tasks via asymmetric mixture-of-transformers, 2026. URL https://arxiv.org/abs/2601.14133.

Yu, S., Yuanchen, X., Du, C., Zhou, Y., Qiu, M., Sun, Q., Zhang, H., and Wu, J. Reverse modeling in large language models. In Chiruzzo, L., Ritter, A., and Wang, L. (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pp. 306–320, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-190-2. doi: 10.18653/v1/2025.naacl-short.27. URL https://aclanthology.org/2025.naacl-short.27/.

Yuksekgonul, M., Bianchi, F., Kalluri, P., Jurafsky, D., and Zou, J. When and why vision-language models behave like bags-of-words, and what to do about it? In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=KRLUvxh8uaX.

Zhang, B., Liu, Z., Cherry, C., and Firat, O. When scaling meets LLM finetuning: The effect of data, model and finetuning method. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=5HCnKDeTws.

Zhang, J., Chen, Y., Xu, Y., Huang, Z., Zhou, Y., Yuan, Y.-J., Cai, X., Huang, G., Quan, X., Xu, H., and Zhang, L. 4d-vla: Spatiotemporal vision-language-action pretraining with cross-scene calibration, 2025a. URL https://arxiv.org/abs/2506.22242.

Zhang, J., Guo, Y., Hu, Y., Chen, X., Zhu, X., and Chen, J. Up-vla: A unified understanding and prediction model for embodied agent. 2025b. URL https://arxiv.org/abs/2501.18867.

Zhang, J., Chen, X., Wang, Q., Li, M., Guo, Y., Hu, Y., Zhang, J., Bai, S., Lin, J., and Chen, J. Vlm4vla: Revisiting vision-language-models in vision-language-action models, 2026. URL https://arxiv.org/abs/2601.03309.

Zhang, L., Liu, Y., Zhang, Z., Aghaei, M., Hu, Y., Gu, H., Alomrani, M. A., Bravo, D. G. A., Karimi, R., Hamidizadeh, A., Xu, H., Huang, G., Zhang, Z., Cao, T., Qiu, W., Quan, X., Hao, J., Zhuang, Y., and Zhang, Y. Mem2ego: Empowering vision-language models with

global-to-ego memory for long-horizon embodied navigation, 2025c. URL https://arxiv.org/abs/2502.14254.

Zheng, R., Liang, Y., Huang, S., Gao, J., III, H. D., Kolobov, A., Huang, F., and Yang, J. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. 2025. URL https://arxiv.org/abs/2412.10345.

# A. Summary of Notations

The notations are described in table 5.

*Table 5.* Summary of Notations

| Notation | Description |
|---|---|
| $f_\theta$ | Vision–Language–Action (VLA) model |
| $f_\theta^V$ | Frozen VLM backbone of the VLA |
| $f_\theta^A$ | Action prediction head |
| $o_t = [I_t, l_t, s_t]$ | Observation at timestep $t$ |
| $I_t$ | RGB image observation (multi-view) |
| $l_t$ | Language instruction |
| $s_t \in \mathbb{R}^{d_s}$ | Robot state vector |
| $a_t$ | Action at timestep $t$ |
| $e_t \in \mathbb{R}^{d_e}$ | VLM embedding of observation $o_t$ |
| $\tau_i$ | Expert trajectory $i$ |
| $\mathcal{D}^{Tr}$ | Set of expert (training) trajectories |
| $\bar{H}$ | Temporal window length for memory chunks |
| $\zeta_t^e$ | Embedding trajectory $(e_{t-\bar{H}+1}, \ldots, e_t)$ |
| $\zeta_t^s$ | State trajectory $(s_{t-\bar{H}+1}, \ldots, s_t)$ |
| $\mathcal{M}_{ext}^{raw}$ | simplified version of memory constructed from demonstrations |
| $\mathcal{M}_{ext}$ | External non-parametric memory built over $\mathcal{M}_{ext}^{raw}$ via appropriate indexing |
| $M$ | Total number of memory entries |
| $\hat{\tau}_t$ | Inference trajectory up to timestep $t$ |
| $\hat{o}_t$ | Observation during inference at timestep $t$ |
| $\hat{e}_t$ | Inference VLM embedding of observation $\hat{o}_t$ |
| $\hat{\zeta}_t^s$ | State trajectory used as memory query |
| $\mathbb{D}(\cdot, \cdot)$ | Distance metric ($L_2$) |
| $\mathcal{I}(\hat{\zeta}_t^s)$ | Indices of top-$k$ retrieved neighbors |
| $\mathcal{E}(\hat{\zeta}_t^s, k)$ | Retrieved embedding set from memory |
| $\mathcal{E}^{avg}$ | Averaged retrieved embedding |
| $\hat{a}_t$ | Action predicted without memory |
| $\hat{a}_t^{mem}$ | Memory-conditioned action prediction |
| $\hat{a}_t^{out}$ | Output action |
| $k$ | top-$k$ in eq 2 |

# B. Real World Experiments

## B.1. Model Architecture and Training Setup

We fine-tune a vision–language–action (VLA) model on a curated dataset of real-world manipulation demonstrations, including picking, placing, and grasping tasks. The model maps multimodal observations—comprising multi-view RGB images and robot proprioceptive state—to temporally extended robot actions. Observations are encoded using a pretrained vision–language backbone augmented with additional visual representations, and projected into a shared latent space through a lightweight fusion module. Proprioceptive inputs are embedded separately and incorporated as conditioning signals. The encoded observations condition a sequence modeling action expert that predicts fixed-horizon joint-space action trajectories for both robot arms. The model is fine-tuned using a diffusion-based objective with conditional flow matching, optimized with AdamW and standard training stabilizers such as mixed-precision training and exponential moving average updates. We use the framework of (Suzuki et al., 2026) for real-world experiments due to its superior performance with low resource requirements.

## B.2. Task Description

For fine-tuning the VLA model, We collect real-world demonstrations on the *Cobot-Magic-AgileX* robot equipped with a 6-DOF arm using a first-person RGB camera setup, with cameras mounted on the front of the robot and the right arm. Visual observations are captured using Intel RealSense D415 cameras at a resolution of $640 \times 480$ and a frame rate of 30 fps. In addition to visual data, robot proprioceptive information is recorded. Demonstrations are collected via joystick-based

teleoperation.

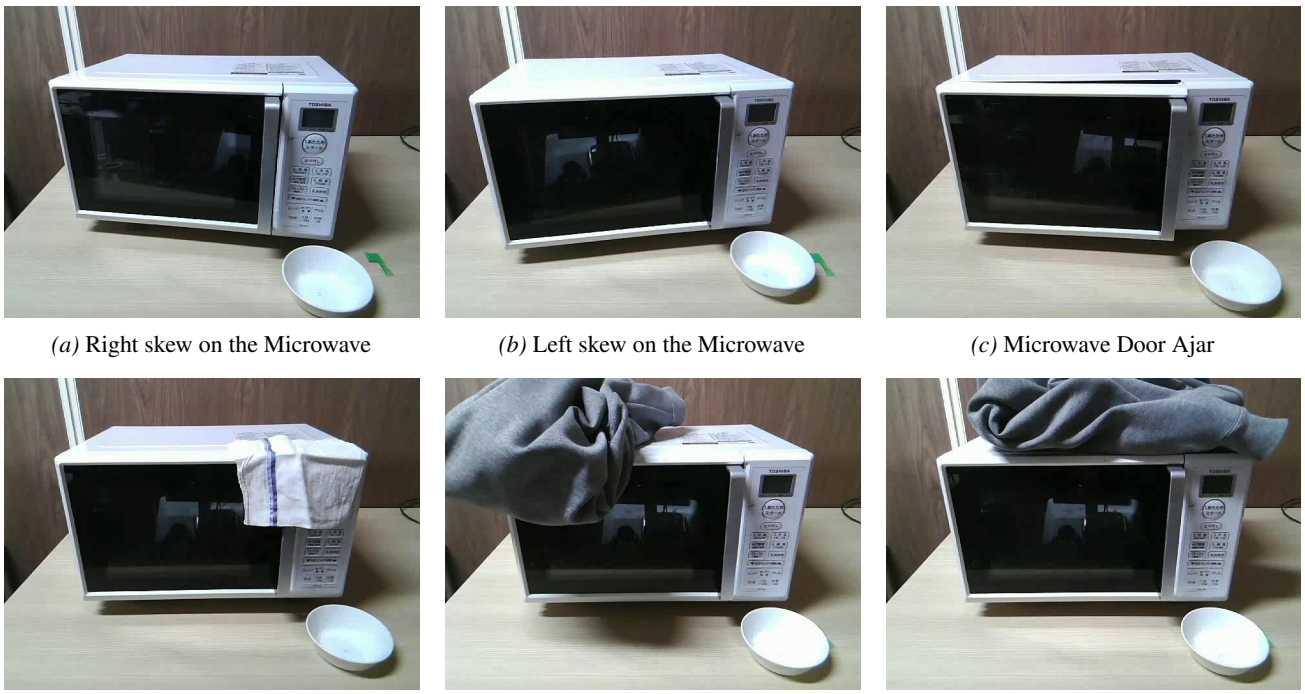
We evaluate the method on three sequential manipulation tasks: (i) Reach Microwave Door, where the robot moves its end-effector to the microwave door handle; (ii) Open Microwave Door, where the robot grasps the handle and opens the microwave door; and (iii) Place Bowl Inside the Microwave, where the robot places a bowl inside the microwave cavity.

Evaluation Protocol. The evaluation follows the step-wise protocol used in CALVIN, where tasks are executed sequentially and success is measured based on the number of subtasks completed in order. Consequently, success rates monotonically decrease as the number of required sequential subtasks increases.

### B.3. OOD Task Variations

To further evaluate generalization under real-world Out-of-Distribution (OOD) settings, we extend the microwave interaction task with a set of environment perturbations introduced during execution. These variations include: (i) Normal configuration (ii) Right Skew on the Microwave (iii) Left Skew on the Microwave (iv) Microwave Door slighlty ajar (v) Cloth Placed on Microwave Handle (vi) Cloth on Top of Microwave (Variant1) (vii) Cloth on Top of Microwav (Variant2)

The evaluation tasks are generated using a mixture of **20% normal configurations** and **80% OOD variations**. Each task is executed for a maximum horizon of **800 steps**. The experiments cover a diverse set of OOD perturbations designed to test robustness to environmental variations.



*(a)* Right skew on the Microwave

*(b)* Left skew on the Microwave

*(c)* Microwave Door Ajar

*(d)* Cloth Placed on microwave Handle

*(e)* Cloth Placed on Top of Microwave (Variant1)

*(f)* Cloth Placed on Top of Microwave (Variant2)

*Figure 4.* Inference Time OOD variations.

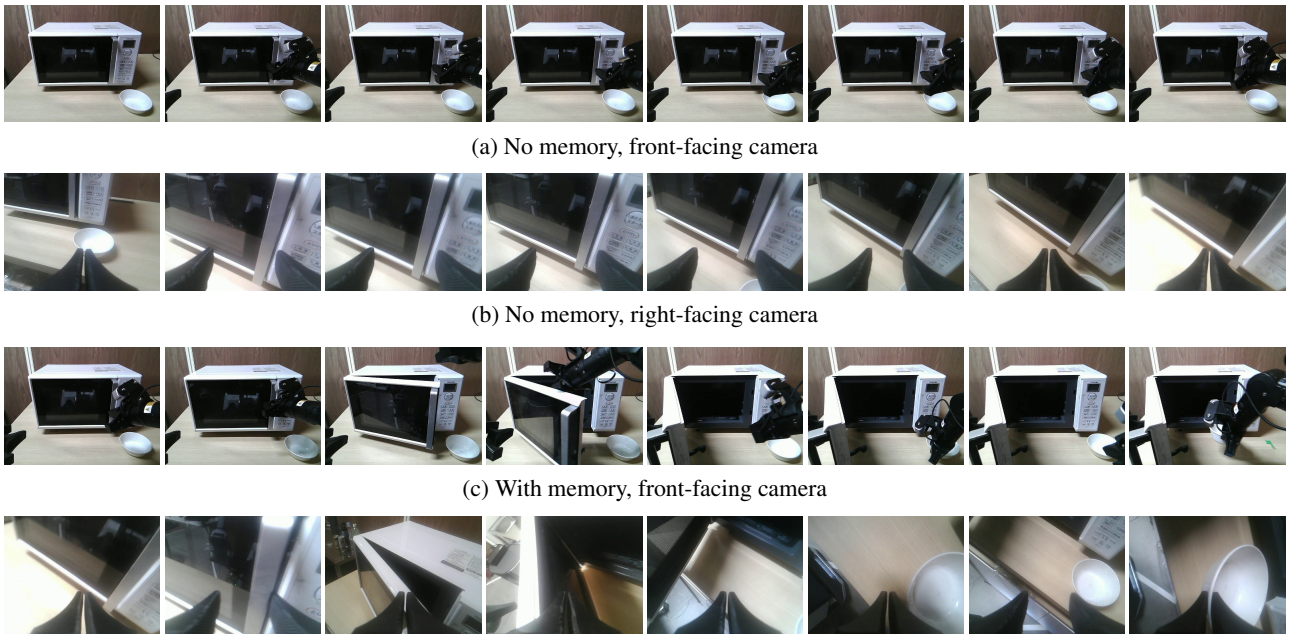
### B.4. Results

Quantitative results are reported in Table 4. We follow a CALVIN-style step-wise evaluation protocol, where a failure at any sub-task results in the remaining tasks being marked as unsuccessful. Consequently, success rates decrease monotonically as the task horizon increases. Qualitative comparisons are shown in Figure 5.

For our long-horizon real world tasks (containing a mix of normal and OOD trials) we observe consistent improvements when incorporating memory. For the first sub-task, *Reach Microwave Door*, the success rate improves from $80\%$ to $\mathbf{90}\%$, corresponding to a relative improvement of $12.5\%$. More notably, the second sub-task, *Open Microwave Door*, shows a

substantial improvement from $30\%$ to $\mathbf{60}\%$, representing a $\mathbf{2}\times$ **increase** in success rate. This improvement is particularly significant because our evaluation protocol includes a large proportion of out-of-distribution (OOD) scenarios, such as skewed microwave positions, partial occlusions, and variations in object placement, which are known to hinder generalization capabilities.

Finally, for the third sub-task, *Place Bowl Inside*, we observe an improvement from $20\%$ to $\mathbf{30}\%$, corresponding to a relative gain of $50\%$. These results suggest that incorporating memory enables the agent to better handle ambiguous observations and maintain task progress under real-world distribution shifts.



(a) No memory, front-facing camera

(b) No memory, right-facing camera

(c) With memory, front-facing camera

(d) With memory, right-facing camera

*Figure 5.* Real World Qualitative comparison of task execution with and without the proposed memory framework. In the default No-Memory (a-b) setting, the gripper arm oscillates between the initial position and the microwave handle, failing to achieve a stable grasp and therefore not progressing to subsequent sub-tasks. In contrast, the proposed Memory framework (c-d) when conditionally activated, successfully reaches and grasps the microwave handle, opens the door, and places the bowl inside the oven, demonstrating consistent task completion.

## C. Experimental Details

All experiments follow the original source-code setup of the respective models. Since our memory construction is completely training-free, all experiments are performed on a single NVIDIA A100 GPU. Memory construction typically requires less than 40 GB of GPU memory, as it only involves a forward pass through the VLM backbone $f_\theta^V$, while inference-time GPU memory usage remains under 6 GB. Table 6 provides the memory-related hyperparameters used in our experiments. The 3 random seeds used are - 0, 67, 469 following seeds used in previous works.

**Robot Configurations:** For the CALVIN suite, the state $s_t$ consists of the robot's proprioceptive information corresponding to the Tool Center Point (tcp) over 8 (thus $d_s = 8$) dimensions with - 3 dims for tcp position in world coordinates, 3 dims for tcp orientation via euler angles, 1 dimension for gripper width and remaining dimensions for gripper binary action of close/open. The action $a_t$ consists of 7 dimensions - 3 dims for tcp position, 3 for the euler angles and 1 for the gripper binary action. For the LIBERO suite, we have $s_t$ consisting of 9 dimensions (thus $d_s = 9$) and actions of 7 dimensions.

### C.1. Inference Overhead due to HNSW Retrieval

In this section, we quantify the inference time overhead introduced by the HNSW retrieval mechanism as a function of the retrieval size $k$ in Equation 2. As discussed in Section 3.2.4, the per-query retrieval complexity is $\mathcal{O}((\log M)\bar{H}d_s + k\bar{H}d_e)$, where the first term corresponds to approximate nearest neighbor search and the second to the aggregation of retrieved

*Table 6.* Memory, $\mathcal{M}_{ext}$ hyperparameters which are kept same across the benchmarks underscoring further efficacy of our framework by reducing tuning overhead. The value of $\bar{H} = 10$ is the default implementation in the respective papers and used verbatim.

| **Hyperparam** | **Value** |
|---|---|
| Memory Size, M | $80,000$ |
| top-$k$ in eq 2 | $10$ |
| Horizon $\bar{H}$ for $\zeta$ | $10$ |

embeddings.

We measure wall-clock retrieval time on a single NVIDIA A100 GPU with $M = 80{,}000$ and $\bar{H} = 10$, averaged over 1000 queries. Figure 6 shows that the retrieval overhead increases only marginally with $k$, remaining nearly constant for smaller values (e.g., $k \leq 20$) and exhibiting a modest rise at larger $k$ (e.g., $k = 25$). Overall, the absolute latency remains low, on the order of $\sim$11–13 ms.

These results demonstrate that HNSW-based retrieval is efficient and scalable, supporting the practicality of always-on memory conditioning without introducing significant inference latency.

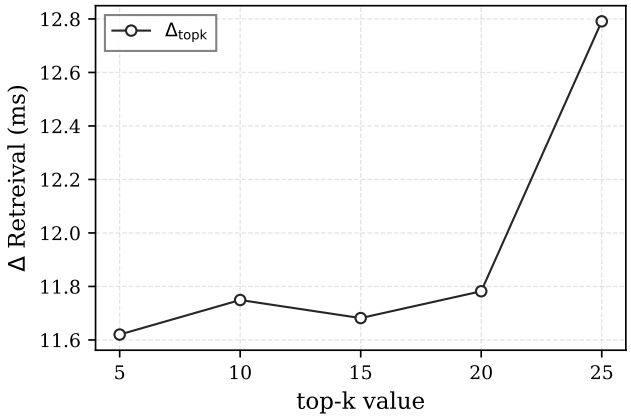



*Figure 6.* Retrieval latency as a function of top-$k$. The overhead remains nearly constant for small $k$ and increases slightly at larger values, while staying within a low-latency regime.

## D. Memory Algorithm

### D.1. Discussion about Memory Conditioning

Here we provide in depth discussion about the $TaskState(\hat{\tau}_t)$ function which is utilized to realize the state of the task given the trajectory being unfolded till timestep $t$. The output of this function essentially characterizes whether the task was completed successfully within acceptable error margins or not. In practice, this is a binary predicate.
In simulated benchmarks, this is often provided by the environment. In contrast, real world systems require explicit onboard sensing and perception modules.

#### D.1.1. Mechanisms for $TaskState(\cdot)$

**Use of Sensors:** Modern robotic platforms are typically instrumented with a suite of sensors including cameras, depth sensors, force/torque sensors, proprioceptive encoders, and tactile modules. These sensors provide continuous feedback about the robot's configuration, interactions with the environment, and external objects. For example, vision and force sensors enable object pose estimation, contact detection, and resistance profiles, which are core to judging task outcomes such as successful insertion or stable placement of an object. Such sensor modalities form the foundation for success detection in real-world systems (Scalise et al., 2019). Below we describe concrete mechanisms for deriving such signals on physical robots and illustrate case-by-case examples

**Learned Success Detectors:** Beyond engineered predicates, explicitly training classifiers that take sensory inputs (vision, proprioception etc) and output a binary success label for the $TaskState(\cdot)$ function. For instance, recent work uses vision–language models as binary success detectors that classify whether a robot observation satisfies a task description, providing a general way to annotate success/failure on real trajectories (Luo, 2024). Such learned detectors generalize across tasks and environments if trained on sufficiently diverse data. Recent models such GPT-5 (Leon, 2026), Qwen-3 (Yang et al., 2025a) have been shown to possess extremely strong capabilities in vision-language perceptions tasks and have been leveraged extensively in the embodied planning literature (Zhang et al., 2025c; Nayak et al., 2025).

**Real World industry examples where sensors are sufficient for task completion status:** We provide references of industrial robots following the aforementioned paradigms here and detail some specific examples below. Industrial Bin-picking, Industrial Packaging, Warehouse Inventory Packaging and such.

#### D.1.2. EXAMPLES

1. **Microwave task:** A robot opens a microwave, places a food bowl inside, presses the start button, and waits for heating to begin. Door opening is detected using force sensing that identifies handle contact and latch disengagement. Bowl placement is verified via a weight sensor embedded within the microwave interior, confirming the presence of the bowl. A successful start is detected using tactile sensing that captures the characteristic force signature of a button press. Task success is inferred purely through sensor-based mechanisms. An example implementation of the corresponding success predicate is shown in the code snippet below.

2. **Industrial robot:** In an industrial assembly task with a 7-DoF arm, such as part insertion and screw tightening, success is inferred from force/torque feedback and vision. Insertion is detected via characteristic wrist force/torque profiles indicating proper seating, while visual pose estimation (eg using YOLO (Jiang et al., 2022)) confirms alignment within tolerance. Screw tightening is deemed complete once the torque controller reaches a specified threshold.

3. **Logistics:** For a mobile manipulator in logistics, success in picking and placing a box onto a conveyor is inferred from multimodal sensing. Grasp success is confirmed through vision and tactile feedback that verify secure holding and force stability. Placement succeeds when vision detects the box (eg using YOLO (Jiang et al., 2022)) within a designated conveyor region and confirms release from the gripper.

```
"""
Task: Open a microwave and heat a food bowl.

Success Mechanism:
This task uses purely sensor-based success detection.
- Door opening is detected via force sensing that identifies
  handle contact and latch disengagement.
- Bowl placement is verified using a weight sensor embedded
  within the microwave interior.
- Successful start is detected via tactile sensing during
  button press.
"""
def microwave_task_success(obs)->bool:
    """
    obs contains:
      - door_open: boolean inferred from force sensing
      - button_pressed: boolean inferred from tactile sensing
      - bowl_weight: estimated weight from microwave weight sensor
    """
    door_ok = obs.door_open

    bowl_placed_ok = (
        obs.bowl_weight >= MIN_EXPECTED_WEIGHT
    )

    button_ok = obs.button_pressed

    return (
        door_ok and
        bowl_placed_ok and
        button_ok
    )
```

**Algorithm 1** Memory Construction and Organization

**Input:** Expert Demonstrations $\mathcal{D}^{Tr} = \{\tau_1, ..., \tau_N\}$, Temporal Horizon $\bar{H}$, Pre-trained VLA Encoder $f_\theta^V$
**Output:** External Memory Index $\mathcal{M}_{ext}$
Initialize simplified (raw) memory storage $\mathcal{M}_{ext}^{raw} \leftarrow \emptyset$
**for** each trajectory $\tau_i \in \mathcal{D}^{Tr}$ **do**
  Let $\tau_i = (o_1^i, \ldots, o_{L(\tau_i)}^i)$
  {Create temporal chunks using sliding window (section 3.2.1)}
  **for** timestep $t = \bar{H}$ **to** $L(\tau_i)$ **do**
    Construct embedding chunk $\zeta_t^e = f_\theta^V((o_{t-\bar{H}+1}, ..., o_t)) = (e_{t-\bar{H}+1}, ..., e_t)$
    Construct state chunk $\zeta_t^s = (s_{t-\bar{H}+1}, ..., s_t)$
    Add $(\zeta_t^e, \zeta_t^s)$ to $\mathcal{M}_{ext}^{raw}$
  **end for**
**end for**
{Organize using HNSW for high recall (section 3.2.1)}
Construct HNSW Index, $\mathcal{H} = HNSW(\{\zeta^s \mid (\zeta^e, \zeta^s) \in \mathcal{M}_{ext}^{raw}\})$ with $\mathbb{D}$ ($L_2$ metric)
$\mathcal{M}_{ext} \leftarrow$ HNSW Index $\mathcal{H}$ along with the $1 \rightarrow 1$ correspondence to $\{\zeta^e \mid (\zeta^e, \cdot) \in \mathcal{M}_{ext}^{raw}\}$
**Return:** $\mathcal{M}_{\text{ext}}$

*Table 7.* Comparison of Data Structures for storage and retrieval, highlighting the necessity for HNSW in our case due to lack of dependence of dimensions and requirements of high recall. $M$ being the memory size, $d = \bar{H} \times d_s$ used as the dimensions. For specific data structures: $L$ is "# of hash tables for LSH", $K$ is clusters in Product Quantization (PQ) and $q$ is the number of candidate comparisons examined after quantization / inverted list probing. Note: the notations used in this table are independent of the notations used in the rest of the paper and are merely for simplicity.

| **Method** | **Data Structure** | **Build Time** | **Query Time** | **Space** | **Recall / Accuracy** | **Best Regime** |
|---|---|---|---|---|---|---|
| Brute Force | None | $O(M)$ | $O(M)$ | $O(Md)$ | Exact | Low/Moderate $d$ |
| KD-Tree | Axis-aligned Binary Tree | $O(M \log(M))$ | $O(M)$ for large $d$ | $O(Md)$ | Exact / Approx | Low $d$ |
| Ball Tree / VP Tree | Metric space tree | $O(M \log(M))$ | $O(\log(M))$ (degrades with $d$) | $O(Md)$ | Exact / Approx | Moderate $d$ |
| LSH | Hash tables (data-independent) | $O(ML)$ | Sublinear | $O(ML)$ | Moderate | High $d$ |
| PQ / IVF-PQ | Quantized codebooks | $O(MK)$ | $O(q \cdot d)$ | $O(M)$ | Good | Low/Moderate $d$ |
| Randon Proj | RP Forests | $O(M \log(M))$ | Sublinear | $O(Md)$ | Moderate | High $d$ |
| HNSW | Hierarchical proximity graph | $O(M \log(M))$ | $O(\log(M))$ | $O(M)$ | High | High $d$ |

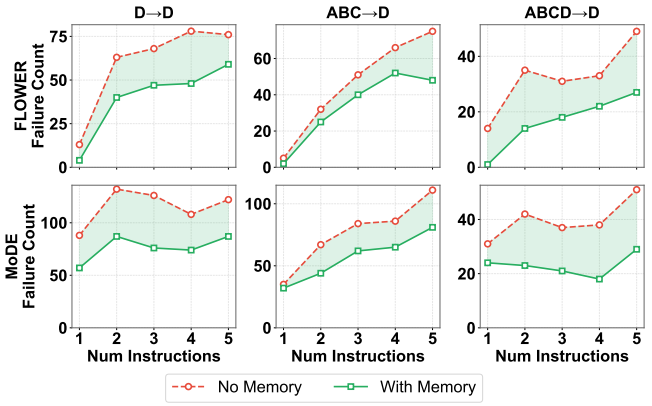



*Figure 7.* Callbacks to the memory. Failure Count (FC) represented by the dashed lines when *memory is not* (FC - No Mem) used against the solid lines when *memory fallback is used* (FC - With Mem) for CALVIN datasets.

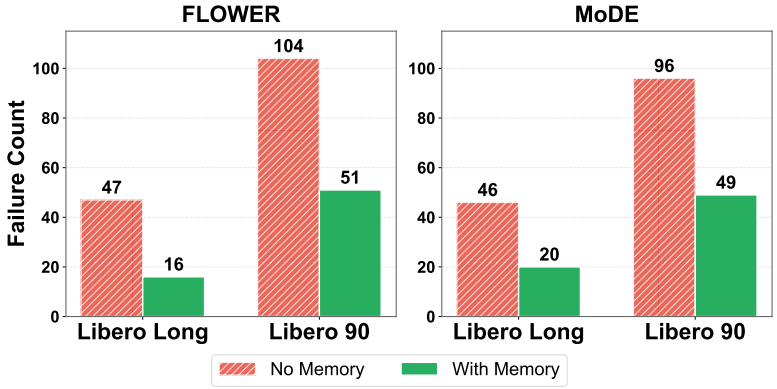



*Figure 8.* Callbacks to the memory. Failure Count (FC) represented by the dashed lines when *memory is not* (FC - No Mem) used against the solid lines when *memory fallback is used* (FC - With Mem) for LIBERO datasets.

## D.2. Analysis

### D.2.1. Memory Callbacks

As discussed previously in section 3, the conditioning on $\mathcal{M}_{ext}$ and the failures it prevent warrants further analysis which is quantitatively highlighted in figure 7 for CALVIN and figure 8 over LIBERO datasets. We plot the Failure Count (FC) for the no-memory and compare it against the FC for memory. Note that the values in figure 7 for a given instruction level $p$ ($p \in \{2, 3, 4, 5\}$) is conditional upon the previous level $p - 1$ and is thus computed only when the task at $p - 1$ is successful. We observe a clear pattern that with increasing level of instruction chaining and thereby the increasing difficulty, the memory callback is more frequent as the failures increase. Noteworthy is the stark reduction in the FC with memory which as low as only $7.2\%$ of the standard no memory case and typically hover at $60 - 70\%$. Similarly, for the LIBERO dataset in figure 8, we have a clear pattern of with memory FC of as low as $34\%$ of the non-memory setting and $44\%$ on average.

### D.2.2. Varying the Memory size $M$ and top-$k$ in eqation 2

An important characterization is the size of the memory $|\mathcal{M}_{ext}| = M$. We plot the variations in figure 9. While it is trivial to note that increasing memory size provides additional gains, but a clear plateau is expected beyond which retrieval time to success rate trade-off is impractical. From this plot, we selected $M = 80,000$ which runs substantially fast, incurring less than $\mathbf{0.01}$ seconds attributed to logarithmic search. This experiment was performed over a held out set of $500$ sequences to select $M$ for the main experiments.
The plots for variation of top-$k$ are discussed in section H.1.

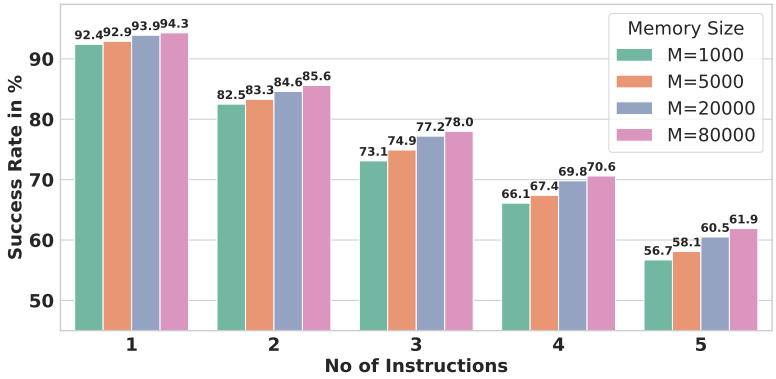
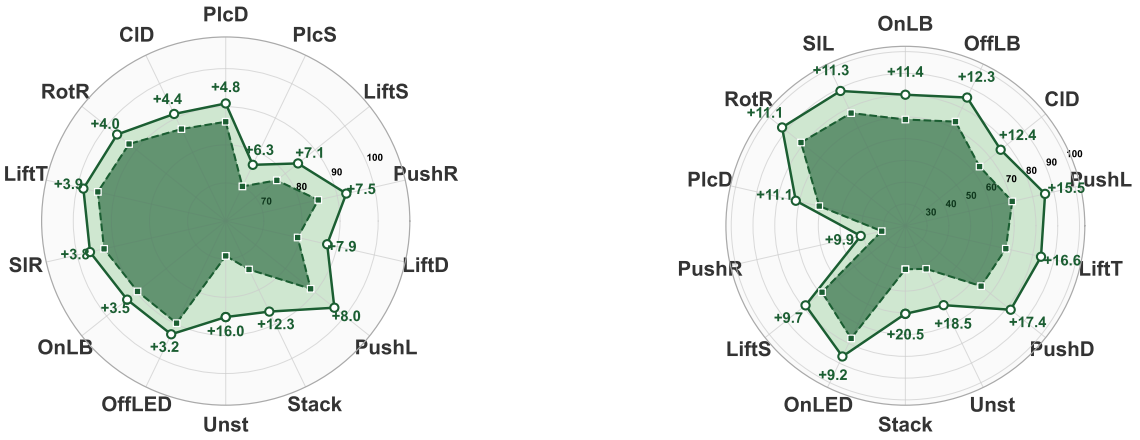



*Figure 9.* Variation of Success Rate across Memory Size $|\mathcal{M}_{ext}| = M$ for the instruction levels.

### D.2.3. Task level results

A deeper analysis showcasing the $\Delta$ improvements in absolute $\%$ attained for each task on CALVIN is detailed in figure 10. Remaining configurations plotted in figure 13 and the discussion for LIBERO is done in section F.2. While the gains are observed uniformly, we observe larger boost in tasks like "Stack/Unstack block", "Move Slider Left/Right", "Push*" where the action sequence can follow previously observed patterns via memory conditioning without substantial dependence on the actual objects characteristics in the environment. Furthermore, we obtain higher improvements on MoDE which has larger margin in the success rates, table 1.




*(a)* FLOWER-VLA

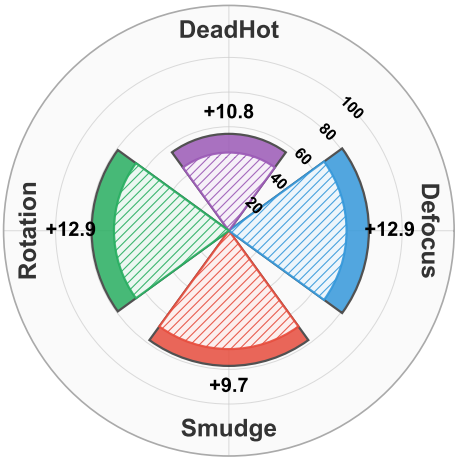
*(b)* MODE-VLA

*Figure 10.* D→D split task-wise $\Delta$ improvements (absolute $\%$), computed as the difference in success rate between the memory-conditioned inference and no memory setup. Task abbreviations in Table 9.

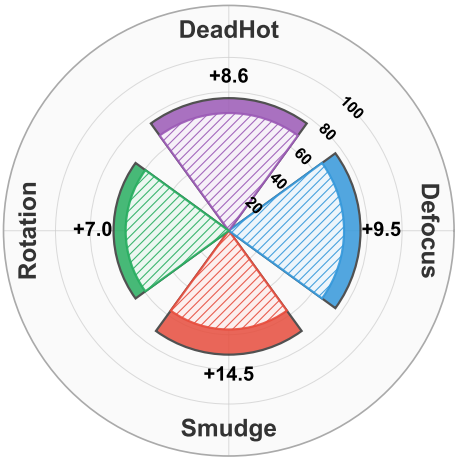
## D.3. Robustness to Corrupted Observations

We further evaluate the robustness of our memory framework under realistic distribution shifts (Wang et al., 2025b) by corrupting the observations $\hat{o}_t$ at inference time. This is done by perturbing all input images in $\hat{I}_t$ at every inference timestep. This setting also captures an important out-of-distribution (OOD) regime in which inference time deviates from the training distribution or where demonstration coverage is sparse in the training data $\mathcal{D}^{Tr}$. We consider the following: Smudge, Defocus, Dead-Hot, and Rotation, as illustrated in figure 12. Quantitative results are shown in figure 11. The hatched regions denote baseline performance without memory conditioning, while the solid regions correspond to our memory-augmented model. Across all corruptions, we observe a consistent degradation in baseline performance, reflecting the known lack of robustness of VLM backbones to visual distribution shifts (Zhang et al., 2026). In contrast, memory-conditioned generation substantially mitigates this degradation by retrieving and leveraging relevant past experiences, even when current observations are corrupted or partially ambiguous.

Notably, robustness gains are amplified under more severe corruptions, where the effective test distribution diverges further from the training data. In these regimes, the memory framework compensates for sparse or imperfect demonstration coverage by grounding action generation in structurally similar past $\zeta^e$ rather than relying solely on brittle visual cues. We observe absolute performance improvements of up to $12.9\%$ on Flower and up to $14.5\%$ on MoDE, with an overall average absolute gain of $10.7\%$, indicating that memory conditioning becomes increasingly beneficial as task difficulty and distribution shift increase.




*(a)* Flower VLA




*(b)* MoDE VLA

*Figure 11.* Comparison of results over observation corruptions (figure E) over LIBERO-long dataset. The hatched results are for *no-memory* case whereas the solid indicates *with memory*. The improvements in bold numbers are absolute $\%$ values. (Table 8 contains corruption parameters.)

## E. Robustness to Corrupted Observations

*Table 8.* Parameters used for the Corruptions used in experiments D.3

| ID | Corruption Type | Key Parameters |
|---|---|---|
| 1 | Defocus Blur | Gaussian Blur with $\sigma = 5$ |
| 2 | Smudge | Number of patches $\sim \mathcal{U}(2, 5)$, patch radius $\in [20, 40]$, blur $\sigma = 20$ |
| 3 | Dead / Hot Pixels | Dead pixel probability $= 0.01$, hot pixel probability $= 0.01$ |
| 4 | Rotation | Rotation angle $\theta \sim \mathcal{U}(-30°, 30°)$ |

In practical deployments, due to various maintenance issues, the equipments undergo degradation over time. We concern with the degradations occuring on the mounted cameras (since language instructions are trivial and no such issues arise) following previous works such as (Li et al., 2025c).

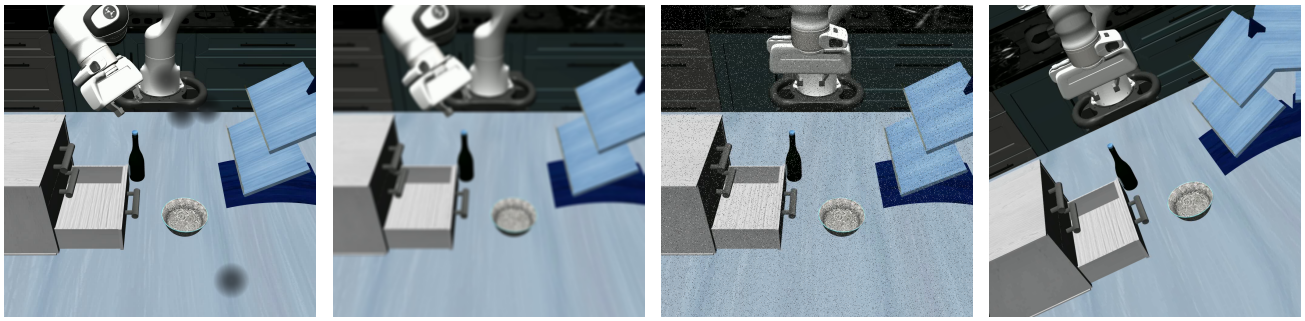
We consider four representative real-world corruptions that frequently arise in deployed vision systems and systematically degrade visual observations:

- *Smudge* models localized optical degradation caused by dirt, grease, or residue on the camera lens. This corruption introduces spatially constrained blur regions that partially occlude scene content while leaving the surrounding areas intact. Unlike global blur, smudge artifacts induce non-uniform information loss and disrupt local texture, edges, and fine-grained details, posing challenges for models.

- *Defocus* simulates out-of-focus imaging by applying a spatially uniform blur across the entire image. This corruption captures scenarios such as incorrect focal length settings or camera autofocus failures. Defocus reduces high-frequency components globally, leading to diminished edge sharpness and contrast throughout the scene, thereby impairing object boundaries and fine structural cues.

- *Dead Hot* represents sensor-level defects by introducing abnormally bright, saturated, or noisy pixels resulting from malfunctioning, overheated, or aging sensor elements. These artifacts manifest as sparse but extreme intensity outliers that do not correspond to the underlying scene content. Such corruptions violate natural image statistics and can disproportionately affect learned representations by introducing spurious high-activation regions that mislead feature extractors and thereby the VLM backbone $f_\theta^V$.

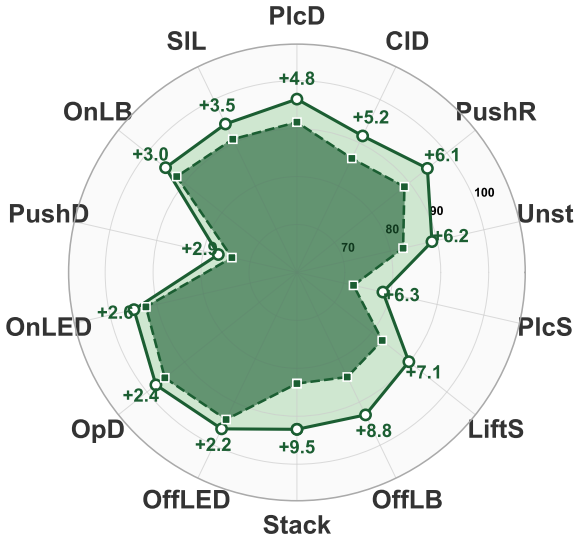
- *Rotation* models camera misalignment or mechanical disturbances by rotating the image by a fixed angle, resulting in a tilted scene orientation. This corruption captures real-world deployment conditions such as improperly mounted cameras or accidental shifts during operation. Rotation disrupts canonical object orientation and spatial priors learned during training.



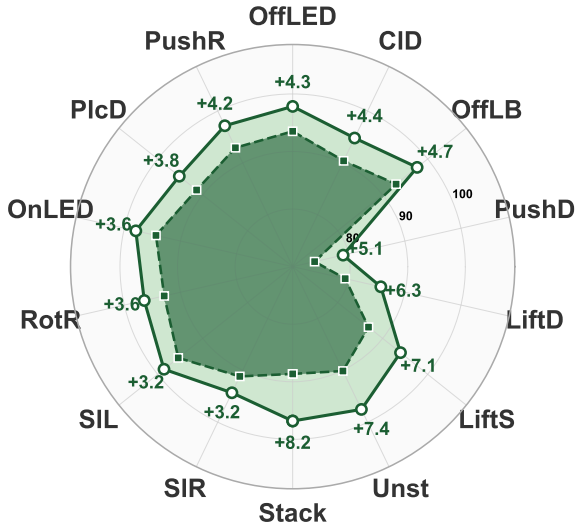
*(a)* Smudge *(b)* Defocus *(c)* Dead Hot *(d)* Rotation

*Figure 12.* Categorization of Corruptions - We induce corruptions on the mounted camera images $I_t$ for observation $o_t$ across the entire inference pipeline.

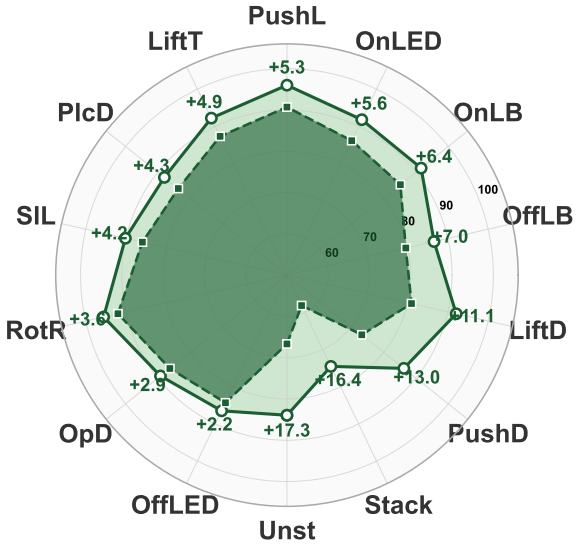
## F. Task Wise Results



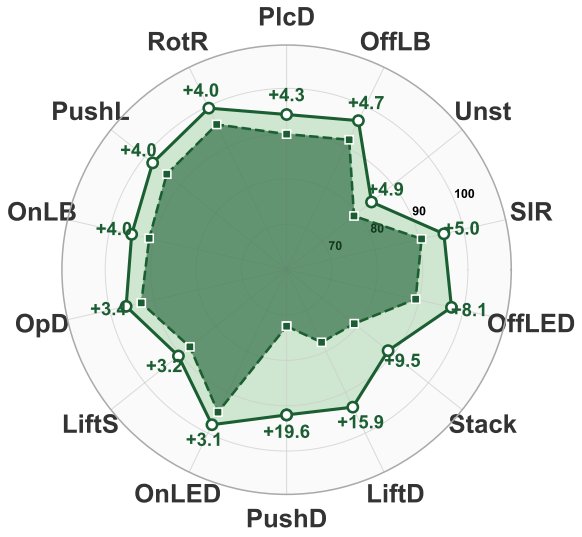

*(a)* Flower: ABC→D




*(b)* Flower: ABCD→D




*(c)* Mode: ABC→D




*(d)* Mode: ABCD→D

*Figure 13.* ABC→D and ABCD→D split task-wise Δ improvements (absolute %), computed as the difference in success rate between memory-conditioned inference and inference without memory for FLOWER-VLA (a,b) and MODE-VLA (c,d). Memory conditioning consistently improves performance across most task groups, with pronounced gains on long-horizon and state-dependent manipulation tasks such as stacking/unstacking (Stack, Unst), drawer interactions (OpD, ClD, PushD, LiftD, PlcD), and multi-stage lifting tasks (LiftS, LiftT). Improvements are generally larger under the more challenging ABC→D split, while gains under ABCD→D are more uniform, indicating *adaptation across domains* when memory is used. Task abbreviations are provided in Table 9.

### F.1. CALVIN

*Table 9.* Task abbreviation mapping to the complete task definitions used in the task-wise CALVIN analysis figures.

| Abbreviation | Task |
|---|---|
| RotR | rotate_block_right |
| RotL | rotate_block_left |
| SlR | move_slider_right |
| SlL | move_slider_left |
| LiftS | lift_block_slider |
| LiftT | lift_block_table |
| LiftD | lift_block_drawer |
| PlcS | place_in_slider |
| PlcD | place_in_drawer |
| PushD | push_into_drawer |
| PushL | push_block_left |
| PushR | push_block_right |
| OpD | open_drawer |
| ClD | close_drawer |
| OnLB | turn_on_lightbulb |
| OffLB | turn_off_lightbulb |
| OnLED | turn_on_led |
| OffLED | turn_off_led |
| Stack | stack_block |
| Unst | unstack_block |

### F.2. LIBERO

Similar to the previous setup for CALVIN, we consider the tasks in the LIBERO-long and LIBERO-90 datasets (Liu et al., 2023). We have three primary categories -

1. **Kitchen:** contains 10 different scenes wherein each scene further contains multiple tasks. Each of these tasks was run 50 times in our experiments 4. As an example, "KITCHEN SCENE 7" contains the following three tasks - " open the microwave" , "put the white bowl to the right of the plate" and "put the white bowl on the plate".

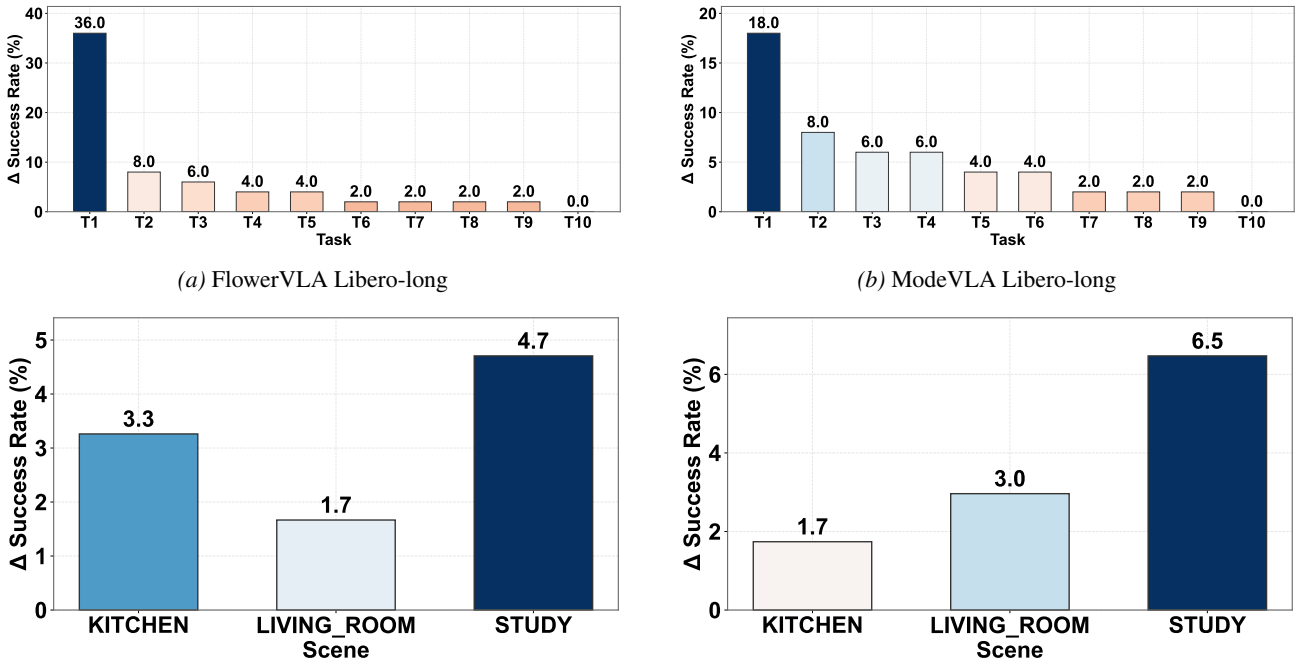
2. **Living Room:** contains 6 scenes.

3. **Study:** contains 4 different scenes.

Since the number of tasks is limited to 10 for LIBERO-long, we plot the task level $\Delta$ improvement in results, whereas for LIBERO-90, we aggregate over the 3 categories for brevity.

*Table 10.* Libero-10 Flower Tasks and Descriptions

| Task ID | Task Name |
|---|---|
| T1 | KITCHEN_SCENE3_turn_on_the_stove_and_put_the_moka_pot_on_it |
| T2 | KITCHEN_SCENE4_put_the_black_bowl_in_the_bottom_drawer_of_the_cabinet_and_close_it |
| T3 | KITCHEN_SCENE6_put_the_yellow_and_white_mug_in_the_microwave_and_close_it |
| T4 | KITCHEN_SCENE8_put_both_moka_pots_on_the_stove |
| T5 | LIVING_ROOM_SCENE1_put_both_the_alphabet_soup_and_the_cream_cheese_box_in_the_basket |
| T6 | LIVING_ROOM_SCENE2_put_both_the_alphabet_soup_and_the_tomato_sauce_in_the_basket |
| T7 | LIVING_ROOM_SCENE2_put_both_the_cream_cheese_box_and_the_butter_in_the_basket |
| T8 | LIVING_ROOM_SCENE5_put_the_white_mug_on_the_left_plate_and_put_the_yellow_and_white_mug_on_the_right_plate |
| T9 | LIVING_ROOM_SCENE6_put_the_white_mug_on_the_plate_and_put_the_chocolate_pudding_to_the_right_of_the_plate |
| T10 | STUDY_SCENE1_pick_up_the_book_and_place_it_in_the_back_compartment_of_the_caddy |




*(a)* FlowerVLA Libero-long

*(b)* ModeVLA Libero-long

*(c)* FlowerVLA Libero-90

*(d)* ModeVLA Libero-90

*Figure 14.* Task-wise Performance Deltas Across Libero Benchmarks.

## G. CALVIN Extension to 6+ Instructions

A key design choice in our extended evaluation protocol is the use of inverse task pairs (e.g., *open drawer* → *close drawer*), requiring the policy to reverse previously executed actions. This is motivated by extensive evidence that sequence reversal poses a fundamental challenge for autoregressive models: the reversal curse (Berglund et al., 2024) shows LLMs trained on "A is B" fail to generalize to "B is A", reverse-direction modeling exposes asymmetric processing in LLMs (Yu et al., 2025; Shafiei et al., 2025), and VLMs exhibit severe order insensitivity (Yuksekgonul et al., 2023) while the Do-Undo benchmark (Mahajan et al., 2025) demonstrates that current VLMs struggle to reverse physical actions. Our inverse-task extension therefore provides a principled stress test of whether memory conditioning can compensate for this inherent directional weakness in VLA policies.

To construct the 6+ instruction sequences, brute force generation and verification as done originally in the CALVIN code did not work due to the exponential search space. We thus resort to an alternative method which operates as follows:to 6+ Instructions

1. **Define Inverse Task Pairs:** We first establish a bijective mapping between tasks that represent semantic and functional opposites (e.g., '(open_drawer, close_drawer)', '(turn_on_led, turn_off_led)'). This creates a set of inverse pairs, where executing one task nullifies the environmental state change caused by the other.

2. **Identify Unresolved Tasks:** For each task sequence $\mathcal{T} = (T_1, T_2, \ldots, T_q)$ ($q = 5$ in the default CALVIN suite as seen from table 1), we identify unresolved tasks. A task is considered unresolved if it is part of an inverse pair but its corresponding inverse does not appear later in the sequence to complete a "cycle". We implement this by processing the sequence chronologically while maintaining a set of unresolved tasks. When a task $T_j$ is encountered, if its inverse is in the set, the inverse is removed (indicating a cycle has been completed); otherwise, $T_j$ is added to the set. Tasks without a defined inverse are ignored.

3. **Generate Candidate Sequences:** After scanning the task sequence $\mathcal{T}$, any remaining tasks in the set are considered unresolved. For each such unresolved task $T_u$, we form a new candidate sequence $\mathcal{T}'$ by appending its inverse, $T'_u$, to the end of the original sequence. This results in a new sequence $\mathcal{T}' = (T_1, T_2, \ldots, T_q, T'_u)$ of length $q + 1$.

*Table 11.* Sample sequences from the CALVIN extension dataset used for evaluation in table 3. Each column describes 1 evaluation sequence.

| Instruction Levels | Seq A | Seq B | Seq C |
|---|---|---|---|
| 1 | push_pink_block_left | push_pink_block_left | push_red_block_left |
| 2 | move_slider_right | move_slider_right | turn_off_led |
| 3 | turn_on_lightbulb | turn_on_lightbulb | move_slider_right |
| 4 | close_drawer | close_drawer | open_drawer |
| 5 | lift_blue_block_slider | lift_blue_block_slider | lift_pink_block_slider |
| 6 | place_in_slider | place_in_slider | place_in_drawer |
| 7 | open_drawer | push_pink_block_right | close_drawer |
| 8 | turn_off_lightbulb | open_drawer | move_slider_left |
| 9 | push_pink_block_right | turn_off_lightbulb | push_red_block_right |
| 10 | move_slider_left | move_slider_left | turn_on_led |

4. **Validate and Augment Dataset:** Each candidate sequence $\mathcal{T}'$ is then validated against its corresponding initial state within the CALVIN simulation environment. This step ensures that the appended action is physically and logically feasible given the state of the environment after the first $q$ tasks, which are already validated to begin with. Only successfully validated sequences are collected, forming our new augmented dataset of horizon length 6.

5. **Extension beyond 6:** The above 4 steps are repeated in order to extend the dataset to horizon (instruction) length up to 10, essentially doubling the $q = 5$ in the original suite.

6. **Lack of Inverse Pair in sequence $\mathcal{T}$:** Additionally, if at any step there are no valid inverse pairs, we append a random subtask $T_{rand}$ and validate it within the environment constraints as above.

This construction has the following advantages:

1. We bypass the brute force generation strategy and keep the resource cost under limits.

2. Within a given sequence $\mathcal{T}'$, we can simultaneously assess the capability of our memory framework in completing the inverse tasks. This is fundamental in various real-world manipulation tasks such as - kitchen tasks, object pick and place etc.

Example sequences are provided in table 11.

## H. Discussion about Retrieval

### H.1. Variation of top-$k$ in eq 2

The plot for the variation of the neighbor parameter $k$ in eq 2 and assigned to 10 in table 6 was selected based on the hold out set results as shown in figure 15.

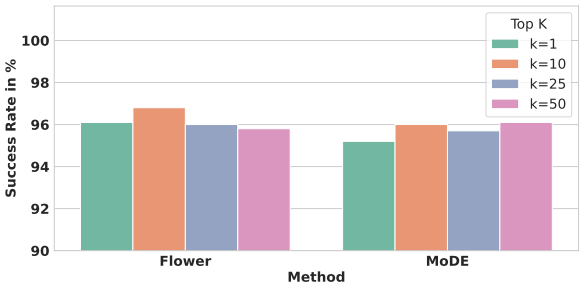



*Figure 15.* Variation of $k$ (eq 2) across the models.

### H.2. Use of VLM embeddings $\zeta^e$ in equation 2

A natural alternative for eq 2 is to condition retrieval on both state information and embeddings of the past $\bar{H}$ timesteps during the rollout of the given trajectory $\hat{\tau}_t$ in the environment. However, this introduces two key challenges. First, combining state and embedding similarities requires careful balancing between their relative contributions, typically through additional hyperparameters while accounting for the fact that the corresponding spaces for $\zeta^s$ and $\zeta^e$ are different. These trade-offs are highly sensitive to the model and dataset, leading to significant tuning overhead and reduced robustness across settings. Second, as discussed earlier, the current VLM backbones used in VLAs are non-robust (Vo et al., 2025; Hancock et al., 2025; Rahmanzadehgervi et al., 2024; Zhang et al., 2026), making them poorly suited for fine-grained retrieval. As a result, directly incorporating embeddings into retrieval can introduce noise rather than useful signal. For these reasons, we rely solely on state-based retrieval, while still leveraging aggregated retrieved embeddings downstream in the action prediction module, where the higher model capacity can better exploit such information.

To further justify this quantitatively, we compute the "Coefficient of Variation" (defined as standard deviation / mean) over the L2 distance across all pairs of embeddings in both – proprioceptive (which is used currently) and VLM backbone embeddings. We observe a spread, measured via the coefficient of variation, of $0.35$ in proprioceptive against a much lower of $0.17$ in visual embeddings, which is half that of the proprioceptive.

## I. Additional Results

### I.1. Comparison against Additional Baselines

*Table 12.* Performance comparison over LIBERO-long for additional baselines. Note that this table is for benchmarking purposes and completeness of the results. In the row corresponding to "Ours", we retain the best numbers from table 2 (borrowed from FlowerVLA). For the baselines, the numbers are borrowed from the respective works. The results describe success rate in $\%$.

| Method | Success Rate |
|---|---|
| TTF-VLA (Liu et al., 2025) | 58.0 |
| TraceVLA (Zheng et al., 2025) | 54.1 |
| 4D-VLA (Zhang et al., 2025a) | 79.1 |
| CronusVLA (Li et al., 2025b) | 86.5 |
| MAP-VLA (Li et al., 2025c) | 83.4 |
| MemoryVLA (Shi et al., 2025) | 93.4 |
| Ours | **96.8** |

*Table 13.* Performance comparison across the CALVIN splits other recent SOTA methods. Note that this table is for benchmarking purposes and completeness of the results. In the row corresponding to "Ours", we retain the best numbers from table 1 (typically fetched from FlowerVLA). For the baselines, the numbers are borrowed from the respective works. The results describe success rate in $\%$.

| Baseline | Split | Level 1 | Level 2 | Level 3 | Level 4 | Level 5 | Avg Seq Len |
|---|---|---|---|---|---|---|---|
| UP-VLA (Zhang et al., 2025b) | ABC→D | 92.8 | 86.5 | 81.5 | 76.9 | 69.9 | 4.08 |
| Uni-VLA (Wang et al., 2025a) | ABC→D | <u>98.9</u> | <u>94.8</u> | 89.0 | 82.8 | 75.1 | 4.41 |
| pi-0 (Black et al., 2024) | ABC→D | 93.8 | 85.0 | 76.7 | 68.1 | 59.9 | 3.84 |
| Ours | ABC→D | **99.8** | **97.3** | **93.3** | **88.1** | **82.3** | **4.61** |
| UD-VLA (Chen et al., 2025) | ABCD→D | <u>99.2</u> | <u>96.8</u> | <u>93.6</u> | <u>90.4</u> | 84.0 | <u>4.64</u> |
| UP-VLA (Zhang et al., 2025b) | ABCD→D | 96.2 | 92.1 | 87.9 | 84.2 | 81.2 | 4.42 |
| Uni-VLA (Wang et al., 2025a) | ABCD→D | 98.5 | 96.1 | 93.1 | 89.9 | <u>85.1</u> | 4.63 |
| Ours | ABCD→D | **99.9** | **98.5** | **96.7** | **94.5** | **91.8** | **4.81** |

### I.2. Comparison of Timesteps for Task Completion

The median timesteps taken, along with the $25-th$ and $75-th$ percentile, to complete the tasks are plotted in figures 17 and 16. Note that for this computation, we consider the total number of timesteps when our memory fallback is activated by adding it to the respective timesteps taken for the task failure in the first run.

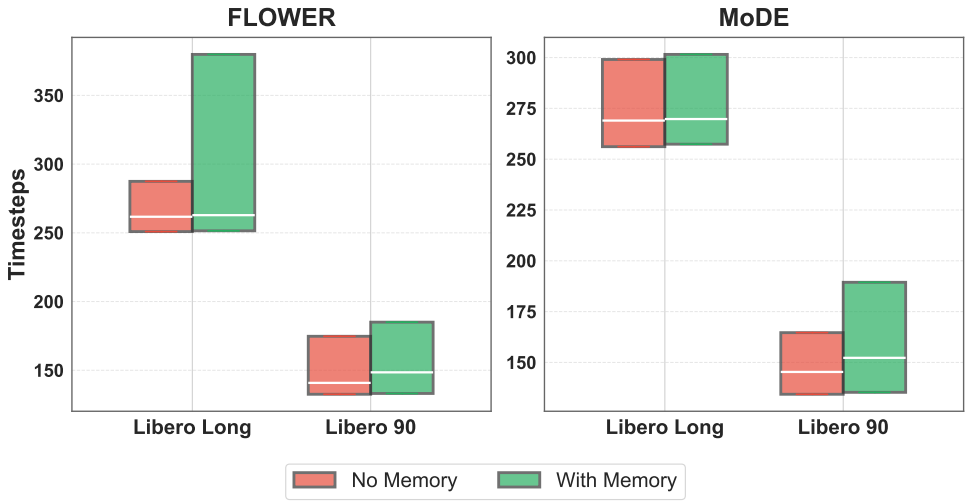



*Figure 16.* Median Time Steps along with the quartiles for the default (No Memory) and with Memory Conditioning on LIBERO datasets. Top of the subfigures denote the dataset split. We obtain substantial gains in success rates without any notable increase in inference timesteps.

| Framework | VLA Training Time Data Access | VLA Inference Time Data Access | Compute |
|---|---|---|---|
| MemoryVLA | Yes | No | 8 A100 GPUs |
| MAP-VLA | Yes | Yes | 6 RTX 6000 Ada GPUs |
| Ours | No | Yes (minimal - Fig. 6 of main paper) | 1 A100 GPU only |

*Table 14.* Comparison of data access and compute across frameworks.

## J. Resource Comparison against MemoryVLA and MAP-VLA

Table 14 details the resource requirements of our framework against notable memory related works.

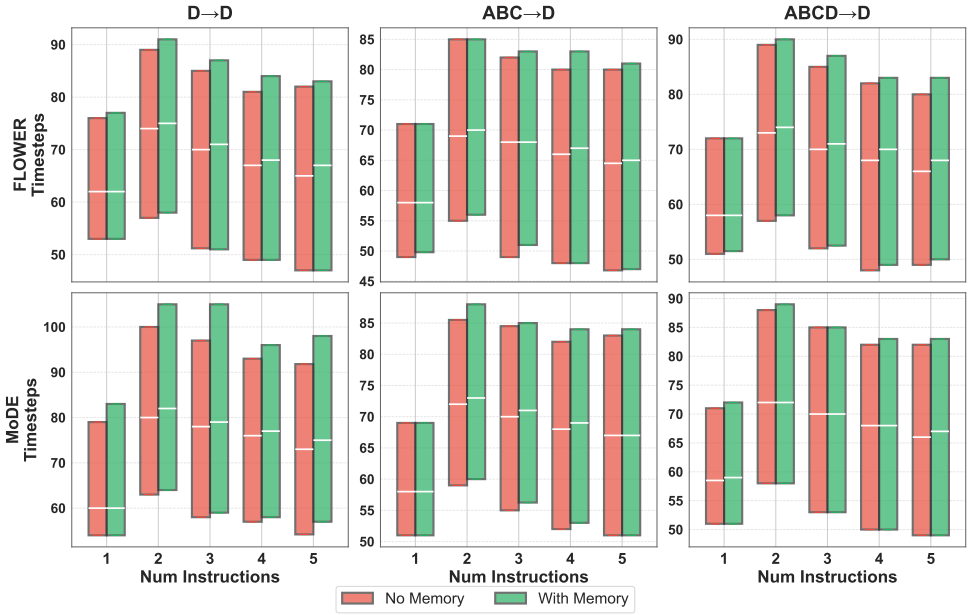



*Figure 17.* Median Time Steps along with the quartiles for the default (No Memory) and with Memory Conditioning on CALVIN datasets. Top of the subfigures denote the dataset split. We obtain substantial gains in success rates without any notable increase in inference timesteps.