SNaP: Self-Narrating Policy with Fast-Weight Context for VLA Control

Anonymous authors
Paper under double-blind review

Abstract

Long-horizon robot manipulation requires recalling past observations and tracking task progress that may not be apparent from the current observation. We introduce SNaP, a self-narrating vision-language-action (VLA) policy that builds context from its own past narrations to guide future decisions. The policy generates short descriptions of its current subtask and relevant task state, using them to condition the current action and accumulating them as context for future narrations and actions. Rather than retaining a growing text history, SNaP encodes past narrations into token-level associations stored in a fixed-size fast-weight narration bank that is updated online during deployment. A separate sensory bank retains proprioceptive context. Readouts from both banks condition subsequent narration and action prediction. The policy thus builds and uses its context online from its own descriptions and observed states. Across three real-robot manipulation tasks requiring object-location recall, cue-based counting, and multi-stage progress tracking, SNaP achieves 90.0% average success, outperforming the strongest evaluated baseline by 23.3 percentage points. Targeted context interventions further show that omitting specific past events changes subsequent subtask predictions, supporting the policy's use of accumulated context.

1 Introduction

Long-horizon manipulation is rarely fully observable. A robot retrieving an object must recall where it was placed before the lid closed; one following a recipe must know how many scoops it has poured and which drink it is preparing. The information that determines the correct action is thus often absent from the current image (Kaelbling et al., 1998). Yet learned manipulation policies, from behavior cloning to recent vision-language-action (VLA) models (Brohan et al., 2023; Kim et al., 2024; Black et al., 2025a; Physical Intelligence et al., 2025), remain reactive: they map the current observation and instruction to an action, and fail once the scene no longer reveals the task state.

Prior work gives the policy history in three ways. The first extends perception in time, widening the observation window (Torne et al., 2025; Jang et al., 2025; Koo et al., 2026) or carrying a recurrent or fast-weight state across steps (Li et al., 2026a;b; Jiang et al., 2026). The second retains selected observations, such as keyframes or feature banks, for later attention (Sridhar et al., 2026; Shi et al., 2026; Mark et al., 2026). Both store history as observations, but control-relevant quantities such as a count, a stage, or a hidden location are abstractions over many frames. The policy must re-derive them at every step or compress them into an opaque state, and longer histories invite copycat behavior (Wen et al., 2020; de Haan et al., 2019). The third stores history as language, appending policy-generated notes or model-written summaries to the prompt (Haresh et al., 2026; Torne et al., 2026; Yin et al., 2026). Language captures the abstract state observations lack, but a growing transcript must be truncated or summarized, often by a separate model, and re-encoded at every step. What is missing is a mechanism by which the policy itself distills the past into a fixed-size task-level state that conditions its subsequent narration and actions. We close this gap by building on the subtask descriptions that policies such as $\pi_{0.5}$ already generate before each action (Physical Intelligence et al., 2025). Such a description names the relevant object, the current operation, and the progress made so far: precisely the state an observation window cannot supply. We give this self-narration a second role: beyond conditioning the current action, each narration is written into a fixed-size fast-weight memory that is read back to condition every future narration and action. Self-narration

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

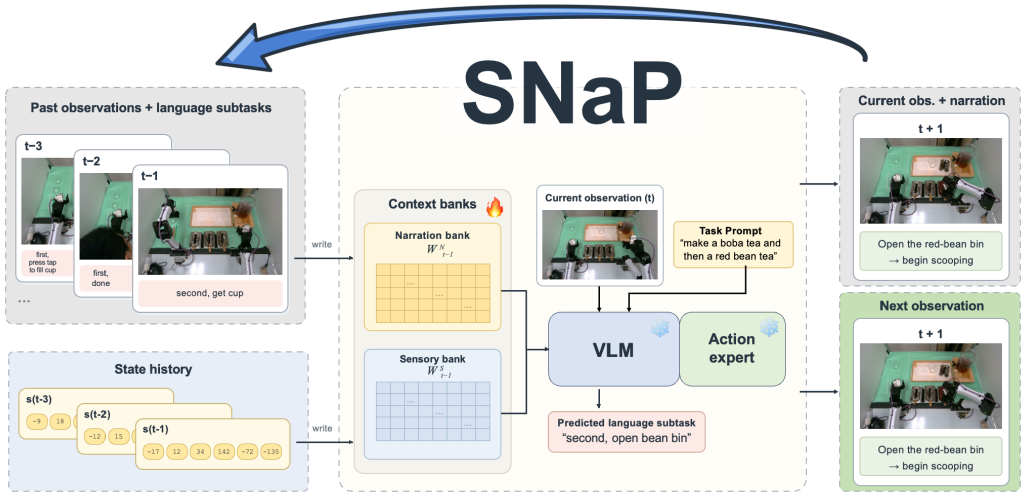

Figure 1: **SNaP overview.** The policy's own past subtask narrations and proprioceptive states are written online into a narration bank W^N_{t-1} and a sensory bank W^S_{t-1}. At tick t, readouts from both banks, the current observation, and the task prompt condition the VLM's subtask narration ("second, open bean bin") and the action expert's action chunk A_t; the new narration and state are then written back for the next tick. Only the two banks change online (flame); all VLM and action-expert weights stay fixed (snowflake).

decides what is remembered; fast weights decide how it persists. The policy thus becomes both author and reader of its own task history, without a separate summarizer or a growing transcript.

We instantiate this idea as **SNaP**, a Self-Narrating Policy with Fast-Weight Context for VLA control (Figure 1). SNaP maintains two episode-level fast-weight banks, one for the policy's own subtask descriptions and one for proprioceptive states. At each step, readouts from both banks join the current observation and instruction to produce a narration and an action chunk, and the new narration and robot state are written back for the next step. Only the banks change online; the VLA weights stay fixed. Training uses subtask-annotated demonstrations to supervise narration and bootstrap memory writes; at deployment, each episode starts from empty banks filled solely by the policy's own outputs. We evaluate SNaP on three real-robot tasks that require recalling a hidden object location, counting an earlier light cue, and tracking progress through a two-drink recipe. SNaP achieves an average success rate of 90.0%, outperforming the strongest baseline by 23.3 points. Targeted interventions further show that omitting specific past events changes subsequent subtask predictions, confirming reliance on accumulated context.

2 Related Work

Historical context for VLA control. Many VLA policies condition actions on the current observation or a short observation window (Kim et al., 2024; Ghosh et al., 2024; Black et al., 2025a; Physical Intelligence et al., 2025). However, tasks involving occluded objects, delayed cues, or repeated actions require information beyond the present frame, as emphasized by recent robotic context benchmarks (Chen et al., 2026; Lei et al., 2026; Dai et al., 2026). Existing approaches extend or compress observation windows (Torne et al., 2025; Jang et al., 2025; Koo et al., 2026), retain selected keyframes or historical features (Sridhar et al., 2026; Mark et al., 2026; Shi et al., 2026), or carry recurrent representations across decision steps (Li et al., 2026a; Cherepanov et al., 2026; Li et al., 2026b). Longer histories can also introduce shortcut reliance on past behavior (de Haan et al., 2019; Wen et al., 2020), motivating designs that consider not only how much history is available, but how it is represented for subsequent control.

Language as task-state context for control. Language provides an intermediate representation for planning, subtask selection, and action generation (Huang et al., 2023; Belkhale et al., 2024;

Zawalski et al., 2025; Physical Intelligence et al., 2025; Shi et al., 2025). SimpleMemVLA uses the hidden states of a subtask generated from video history to condition its action head, recomputing this representation at each decision (Yin et al., 2026). Other methods retain language across decisions: Notes-to-Self appends policy-generated notes to the prompt (Haresh et al., 2026), while MEM trains a high-level policy to update natural-language summaries using LLM-written summary labels (Torne et al., 2026). MemER also evaluates predicted subtask histories as textual context alongside or instead of visual keyframes (Sridhar et al., 2026). These approaches motivate using language to express historical task information. They differ, however, in whether that information is regenerated from retained observations or carried forward as text, whose length must be managed through selection or summarization.

Fixed-size context compression. Compact context representations have been studied through prompt compression (Mu et al., 2023; Chevalier et al., 2023; Ge et al., 2024) and recurrent or segment-level compression (Rae et al., 2020; Bulatov et al., 2022; Munkhdalai et al., 2024). Fast-weight models instead encode associations in an evolving parameter state, using delta-rule or test-time learning updates (Schlag et al., 2021; Yang et al., 2024; Sun et al., 2025; Behrouz et al., 2025). RoboTTT brings fast-weight layers to robot policies to retain long visuomotor context (Jiang et al., 2026). SNaP connects this fixed-capacity formulation with self-narration: it maintains contextual associations derived from the policy's narration tokens and a separate stream of proprioceptive observations in two independent banks. Each bank uses a learned nonlinear address map, while online adaptation updates only its output matrix. Retrieved context conditions both subsequent narration and action prediction, allowing the policy's own task-state descriptions to inform future control without retaining an expanding text transcript or using a separate summarization model.

3 SNAP

3.1 PRELIMINARIES

Narration-Conditioned VLA Policies. We build on a narration-conditioned VLA policy derived from $\pi_{0.5}$ (Physical Intelligence et al., 2025). At decision tick t (one policy step), the policy receives an observation $o_t = (I_t, p_t)$, a task instruction ℓ, and retrieved context tokens R_t, where I_t denotes the camera images and p_t the normalized proprioceptive state. It first generates a self-narration y_t, a short description of the current task state, and then predicts an action chunk A_t, a sequence of robot control actions. Let θ denote all learned slow parameters of the policy and context modules. The narration and action heads, π_θ^{nar} and π_θ^{act}, are conditioned as

$$y_t \sim \pi_\theta^{\text{nar}}(\cdot \mid o_t, \ell, R_t), \qquad A_t \sim \pi_\theta^{\text{act}}(\cdot \mid o_t, \ell, R_t, y_t). \tag{1}$$

Both heads share one VLA backbone and its parameters θ. SNaP supplies R_t from context accumulated at earlier ticks, so that past information can guide both narration and action prediction.

Fast-Weight Context. Fast weights carry sequence context in parameters that evolve during execution (Schlag et al., 2021). We treat each fast-weight bank as an associative memory: a learned address network maps a write key or a read query into an address space, and a fast-weight matrix maps the resulting address feature to a value.

Let d_k, d_h, and d_v denote the key/query, address-feature, and value dimensions; all feature vectors are columns. Given an input $x \in \mathbb{R}^{d_k}$, a key during writing or a query during reading, the address network $\phi_\theta : \mathbb{R}^{d_k} \to \mathbb{R}^{d_h}$ produces an address feature $\phi_\theta(x)$. For a read query q_t, the fast-weight matrix $W_{t-1} \in \mathbb{R}^{d_h \times d_v}$ returns the raw readout $r_t \in \mathbb{R}^{d_v}$,

$$r_t = \text{Read}(W_{t-1}, q_t) = W_{t-1}^\top \phi_\theta(q_t). \tag{2}$$

The address network is trained offline and frozen at deployment; only the fast-weight matrix changes online, through the delta rule described in Sec. 3.2. SNaP maintains two independent banks, indexed by $b \in \{N, S\}$: the narration bank retains self-narration context and the sensory bank retains proprioceptive context. Each has its own fast-weight matrix W_{t-1}^b and address network $\phi_{\theta_b}^b$, where $\theta_b \subset \theta$ denotes the bank-specific slow parameters.

3.2 FAST-WEIGHT CONTEXT FROM SELF-NARRATION AND PROPRIOCEPTION

Narration Bank. The narration bank stores token-level key–value pairs: each key encodes the textual context preceding a narration token, and each value encodes the token itself. Let an accepted

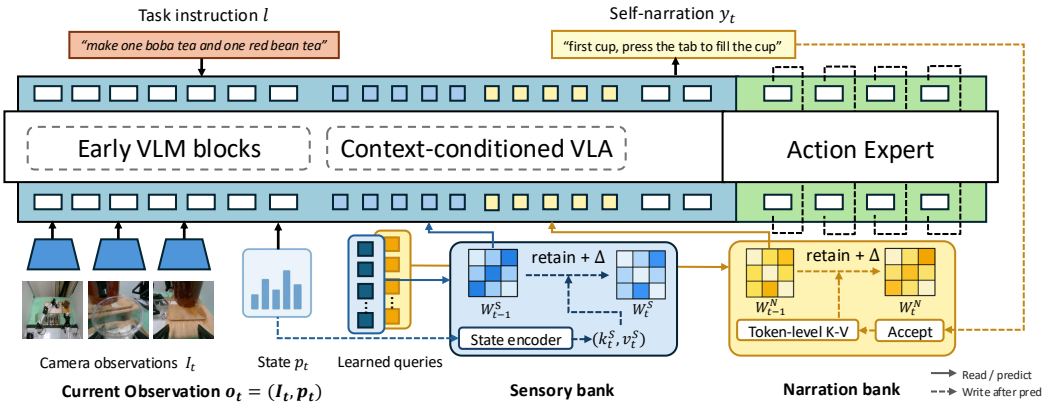

Figure 2: Overview of SNaP. Retrieved context conditions the policy's narration and action predictions. Accepted narrations provide token-level key–value pairs to the narration bank, and proprioceptive states provide one key–value pair per tick to the sensory bank. Only the two fast-weight matrices change online; their readouts at later ticks supply context for future decisions.

narration be $y_t = (x_{t,0}, \ldots, x_{t,L_t-1})$ with L_t valid tokens. We re-encode y_t with a causal, text-only pass that shares the policy's token embedder and early Transformer blocks, and let $h^{\text{ctx}}_{t,j}$ denote the causal hidden state preceding $x_{t,j}$; the first token reuses the first available causal state (Appendix A). The *key encoder* f^N_K produces the write key $k^N_{t,j} \in \mathbb{R}^{d_k}$:

$$k^N_{t,j} = f^N_K\left(\text{sg}(h^{\text{ctx}}_{t,j})\right), \tag{3}$$

where sg denotes stop-gradient. Because the key encodes context rather than the token, the same token is written under different keys when its preceding text differs.

The value encodes the token itself. With E the shared token embedder and f^N_V the *value encoder*, the write value $v^N_{t,j} \in \mathbb{R}^{d_v}$ is

$$v^N_{t,j} = f^N_V(\text{sg}(E(x_{t,j}))). \tag{4}$$

Each valid token thus contributes one pair $(k^N_{t,j}, v^N_{t,j})$. Both encoders combine learned projections with feature normalization and remain trainable, whereas the shared backbone features are detached on this write path. Reference statistics and regularized key whitening are detailed in Appendix A.

Sensory Bank. The sensory bank retains the robot's proprioceptive history. A learned *state encoder* f_{state} maps the normalized state p_t to an intermediate representation $u_t = f_{\text{state}}(p_t)$, from which the bank's key encoder f^S_K and value encoder f^S_V produce a state-dependent pair $k^S_t \in \mathbb{R}^{d_k}$, $v^S_t \in \mathbb{R}^{d_v}$:

$$k^S_t = f^S_K(u_t), \qquad v^S_t = f^S_V(u_t). \tag{5}$$

The sensory bank writes one pair at every valid (i.e., non-padded) tick, whether or not the narration bank accepts a new narration. Implementation details are in Appendix A.

Online Fast-Weight Updates. Given a key–value pair (k, v) for bank b, we compute its address $z = \phi^b_{\theta_b}(k)$ and the value the bank currently predicts at that address, $\hat{v} = (W^b)^\top z$. A normalized delta rule then corrects the prediction toward v:

$$W^b \leftarrow W^b + \eta \frac{z(v - \hat{v})^\top}{\|z\|_2^2}, \tag{6}$$

with update strength η. This rank-one update modifies the memory specifically at the addressed key, rather than simply accumulating the incoming value; with $\eta = 1$, as used here, the readout at z becomes exactly v after the update.

A narration y_t is accepted for writing if it is nonempty, differs from the most recently accepted narration $\bar{y}_{t-1}$, and is decoded with a mean selected-token probability of at least 0.9; we write $c^N_t \in \{0, 1\}$ for this decision. At each valid tick, both banks are first decayed by a fixed retention factor, $W^b \leftarrow \rho W^b$ with $\rho = 0.99$. The sensory bank then writes its proprioceptive pair and, if

$c_t^N = 1$, the narration bank writes the token pairs of y_t sequentially. A rejected narration writes nothing, and padded ticks leave both banks unchanged. The exact rule and further details are in Appendix A.

3.3 Context Readout and Policy Conditioning

Conditioned Context Queries. Let H_t denote the hidden representations of the valid text-prefix positions after the early VLA blocks; through multimodal attention, these already incorporate the current image. Each bank issues $K = 8$ read queries per tick from learned query bases B_i^b, $i \in \{1, \dots, K\}$. The bank's *query generator* $\mathcal{Q}_b$ conditions each base on pooled features of H_t and an encoding of $\bar{y}_{t-1}$, refines it by cross-attention to H_t, and projects and normalizes the result (Appendix A). The resulting query $q_{t,i}^b \in \mathbb{R}^{d_k}$ is read through the same address network and fast-weight matrix used for writing:

$$\begin{aligned} q_{t,i}^b &= \mathcal{Q}_b\big(H_t, \bar{y}_{t-1}; B_i^b\big)\,, \\ r_{t,i}^b &= (W_{t-1}^b)^\top \phi_{\theta_b}^b(q_{t,i}^b), \end{aligned} \tag{7}$$

where $r_{t,i}^b \in \mathbb{R}^{d_v}$ is the raw readout. The two banks have independent query generators. Because K is fixed, the readout cost per tick is constant, however many pairs have been written during the episode.

Conditioning Narration and Action. Each raw readout $r_{t,i}^b$ is scale-calibrated and gated, then combined with a learned read-position embedding to form a context token $m_{t,i}^b$. Concatenating the K tokens from each bank gives R_t, a set of $2K$ context tokens. We insert them into the VLA prefix at the same intermediate-layer boundary from which the queries are built, so only the subsequent blocks see retrieved context. Context tokens attend only to one another, whereas narration and action tokens may attend to the context tokens. The action expert also attends to the current narration tokens, which implements the conditioning in Eq. 1. Readout calibration, context insertion, and attention masks are detailed in Appendix A.

3.4 Training and Online Execution

Sequence Training. We initialize SNaP from a narration-conditioned base policy and optimize it with flow matching on continuous actions and cross-entropy on narration tokens and discretized FAST action tokens (Pertsch et al., 2025), following $\pi_{0.5}$:

$$\mathcal{L} = \mathcal{L}_{\text{flow}} + \lambda_{\text{CE}} \mathcal{L}_{\text{token}}. \tag{8}$$

Training proceeds in two stages that differ only in what the narration bank writes: first the annotated narrations, then the policy's own greedy predictions computed under teacher forcing. In both stages the action expert is conditioned on annotated narrations and the sensory bank writes the observed states. We train on short chronological windows and prefill both banks from bounded histories, annotated narrations for the narration bank and proprioceptive states for the sensory bank, which avoids unrolling the entire preceding trajectory through the policy. Within a window we use truncated back-propagation through time, carrying the fast-weight states forward while bounding activation storage. Sequence sampling, context prefill, and optimization details are in Appendix B.

Online Execution. At deployment both banks start empty, $W_0^N = W_0^S = 0$, and $\bar{y}_0$ is the empty narration. Each tick first computes H_t from the current observation and instruction. Let Read_θ denote the two-bank readout of Sec. 3.3, Decode_θ greedy autoregressive decoding from π_θ^{nar}, and Update the retention and gated writes of Sec. 3.2. One tick proceeds as

$$\begin{aligned} R_t &= \mathrm{Read}_\theta(W_{t-1}^N, W_{t-1}^S; H_t, \bar{y}_{t-1}), \\ y_t &= \mathrm{Decode}_\theta(o_t, \ell, R_t), \\ A_t &\sim \pi_\theta^{\text{act}}(\cdot \mid o_t, \ell, R_t, y_t), \\ (W_t^N, W_t^S) &= \mathrm{Update}(W_{t-1}^N, W_{t-1}^S; y_t, p_t, c_t^N), \end{aligned} \tag{9}$$

where c_t^N is computed from y_t and $\bar{y}_{t-1}$ as above. If the narration is accepted, $\bar{y}_t = y_t$; otherwise $\bar{y}_t = \bar{y}_{t-1}$. The online context is thus built entirely from the policy's own narrations and the observed states, with all slow parameters frozen.



Figure 3: **Real-robot manipulation tasks.** Representative top-camera frames illustrate (a) Hidden Object Retrieval, (b) Cued Bean Scooping, and (c) Two-Drink Recipe. Frames follow temporal order from left to right, continuing onto the next row in (c).

4 Experiments

We evaluate SNaP on real-robot manipulation tasks whose correct behavior depends on context accumulated earlier in the episode. After describing the tasks and protocol, we address three questions:

1. When does self-narrated fast-weight context improve long-horizon VLA control?
2. How should task context be represented and stored?
3. Does the policy rely on task-relevant information in its accumulated context?

4.1 Tasks and Evaluation Setup

We design three real-robot manipulation tasks that exhibit different forms of temporal dependence, with episodes lasting up to five minutes. The robot must recall previously observed information, track progress through multi-stage behavior, and carry earlier cues across repeated actions. Figure 3 shows the three tasks, their instructions, and representative stages of execution.

Hidden Object Retrieval. The robot watches objects being placed into three lidded bins and must later open the bin containing a specified target, given an instruction such as "find the banana." Once the lids are closed, the objects are no longer visible, so selecting the correct bin requires recalling where the target was placed. In the example in Figure 3(a), the banana is placed in the third bin while other objects occupy the remaining bins; the robot later reopens the third bin to reveal the target. The task separates the observation of task-relevant evidence from the action that depends on it.

Cued Bean Scooping. The robot is instructed to "scoop the beans into the tray as many times as the green light blinked." A sequence of green flashes specifies the number of transfers, and a subsequent

yellow light signals the start. The robot must perform that many scoop-and-pour cycles, then put down the scoop and return to rest. Figure 3(b) shows three green flashes followed by three scooping cycles. The count is supplied only by the earlier visual cue, and successive cycles look alike, so the policy must retain the target while tracking how many transfers it has completed. The task therefore combines cue recall with progress-dependent decisions about when to stop.

Two-Drink Recipe. The robot follows the instruction "make a boba tea, then a red bean tea." It first observes the locations of sago, boba, and red beans in three ingredient bins, then prepares the drinks in order. For each drink, it obtains a cup, opens the relevant bin, transfers the required number of scoops, returns the scoop, closes the bin, positions the cup at the dispenser, and presses the tap to fill it. In the recipe shown in Figure 3(c), the first drink requires three scoops of boba and the second requires one scoop of red beans. Cup handling, scooping, and filling recur across the two drinks, but the ingredient and required count change. The task combines spatial recall, counting, and multi-stage sequencing within one extended episode.

Evaluation Setup. All methods run on a dual-arm YAM robot, with each arm providing 6-DoF manipulation and one gripper degree of freedom. Demonstrations are collected using the GELLO teleoperation framework (Wu et al., 2023); we train on 70, 80, and 160 demonstrations for Hidden Object Retrieval, Cued Bean Scooping, and Two-Drink Recipe, respectively, and all methods are trained on the same demonstrations. The policy receives RGB observations from one third-person camera and two wrist-mounted cameras, resized to 224×224 with aspect-ratio-preserving padding, and runs at 30 Hz with 50-step action chunks (approximately 1.67 s of control). Closed-loop execution uses training-time real-time chunking (Black et al., 2025b) with a delay budget of 15 control steps and a nominal replanning interval of five control steps. We use 30 real-robot trials per task for the success-rate evaluations in Q1–Q2, and 15 paired queries per intervention for Q3.

4.2 Results

Q1: When does self-narrated fast-weight context improve long-horizon VLA control? We compare SNaP with the base $\pi_{0.5}$ policy (Physical Intelligence et al., 2025), MemoryVLA (Shi et al., 2026), a $\pi_{0.5}$ variant conditioned on past predicted subtask sentences, and MemER (Sridhar et al., 2026), which retains selected visual keyframes. SNaP uses the narration bank and sensory bank described in Sec. 3.2, both with nonlinear address networks. As shown in Figure 4, SNaP achieves the highest average success rate at 90.0%, compared with 66.7% for MemER and 54.4% for past-sentence conditioning.

The task-level results show where this advantage arises. On Hidden Object Retrieval, SNaP and MemER both achieve 100% success, showing that both retain the target-location information this task requires. The gap is larger on Cued Bean Scooping, where SNaP reaches 93.3%, compared with 50.0% for past-sentence conditioning and 40.0% for MemER. This task requires interpreting a brief sequence of flashes and then tracking subsequent scooping progress, rather than recalling a single object location. On Two-Drink Recipe, SNaP achieves 76.7%, compared with 60.0% for MemER and 36.7% for past-sentence conditioning. Here, the policy must retain ingredient locations while updating the active drink, scoop count, and task stage. The base policy and MemoryVLA rarely complete either counting task. SNaP is therefore particularly beneficial when control requires both recall of earlier evidence and an evolving representation of task progress.

Q2: How should task context be represented and stored? We ablate two design choices: the address network of each fast-weight bank, and which historical context is stored alongside self-narration (Table 1). Every variant includes the narration bank. In SNaP, the address network of each bank is nonlinear (an MLP with three hidden layers); the *linear* variant removes the hidden layers, leaving a linear map into the address space. For stored context, we optionally add sensory bank (proprioceptive history), a *visual* bank, or both. Each variant still receives the current images and proprioceptive state as policy inputs; the ablation concerns only which history is written to memory.

The nonlinear address network outperforms its linear counterpart on every task under every context configuration, raising average success by 30.0 to 38.9 points. With the narration bank alone, average success rises from 44.4% to 83.3%; with the narration and sensory banks, from 51.1% to 90.0%. These comparisons support the nonlinear address network, although they do not isolate whether the gain comes from nonlinearity, added capacity, or both.

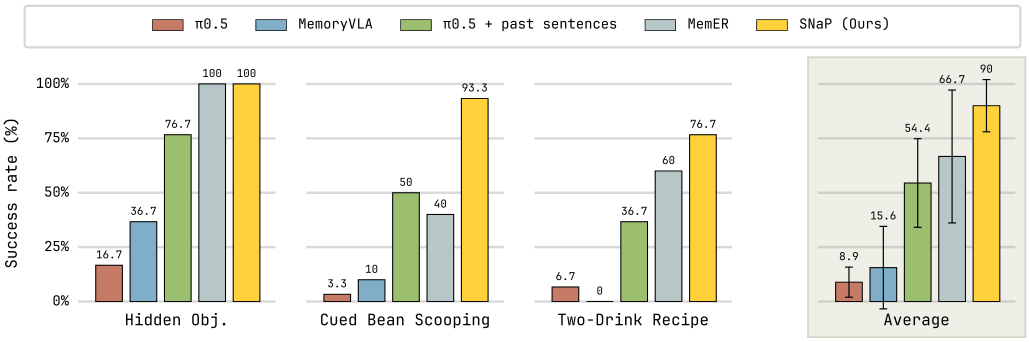

Figure 4: **Baseline comparison across three real-robot tasks.** Success rates over 30 trials per task on Hidden Object Retrieval, Cued Bean Scooping, and Two-Drink Recipe, together with the average across tasks. Error bars on the average panel show the standard deviation of the per-task success rates.

Table 1: **Address-network and stored-context ablations.** All variants include the narration bank. Visual and Sensory indicate whether visual history and proprioceptive history are also written to memory; every variant still observes the current images and state. Entries are successes out of 30 trials; Avg. is the mean success rate across tasks. Bold marks the best entry in each column, and the shaded row is the SNaP default.

Address network	Visual	Sensory	Hidden Obj.	Bean Scoop.	Two-Drink	Avg. (%)
Linear	–	–	22/30	10/30	8/30	44.4
	✓	–	23/30	11/30	8/30	46.7
	–	✓	23/30	13/30	10/30	51.1
	✓	✓	21/30	15/30	9/30	50.0
Nonlinear (3 hidden layers)	–	–	29/30	26/30	20/30	83.3
	✓	–	28/30	25/30	18/30	78.9
	–	✓	**30/30**	28/30	**23/30**	**90.0**
	✓	✓	27/30	**29/30**	16/30	80.0

Storing more history is not always better. Adding the sensory bank to narration-only context improves all three tasks under both address networks and raises average success by 6.7 points in each case, indicating that proprioceptive history complements narration. Visual history does not help: added to the nonlinear narration-plus-sensory configuration, it improves Cued Bean Scooping from 28/30 to 29/30 but reduces Two-Drink Recipe from 23/30 to 16/30, lowering the average from 90.0% to 80.0%. We use the nonlinear narration bank and sensory bank as the SNaP configuration in the main comparisons; this combination achieves the highest observed average across tasks.

Q3: Does the policy rely on task-relevant information in its accumulated context? Task success alone does not establish whether later decisions depend on specific historical information. We therefore perform targeted context interventions on the same trained SNaP policy without retraining. For each intervention, we reconstruct the accumulated context while omitting one task-relevant earlier event, keeping the current observation, task instruction, and policy parameters unchanged. We evaluate 15 paired queries per intervention, comparing the subtask predicted in the self-narration against the same ground-truth subtask under full and event-removed context. Figure 5 illustrates the three interventions and reports their subtask accuracies, rather than full-task success rates.

For Hidden Object Retrieval, removing the banana-placement event reduces target-bin accuracy from 100% to 26.7%. For Cued Bean Scooping, removing one green blink reduces accuracy at the yellow go cue from 93.3% to 0%; in the illustrated query, the predicted target drops from three scoops to two. For Two-Drink Recipe, removing one completed boba scoop also reduces accuracy from 93.3% to 0%; the illustrated prediction requests another scoop instead of closing the bin. In each case, omitting a single earlier event changes the later prediction in the direction that event

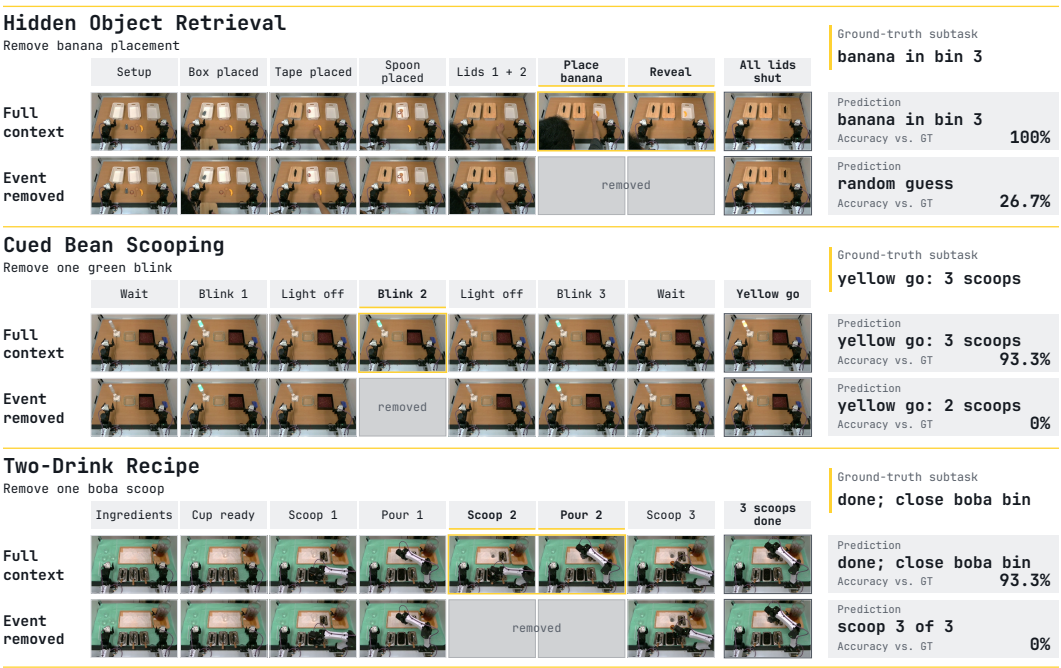

Figure 5: **Targeted context interventions.** Full and event-removed context are compared using the same policy, query observation, and instruction. Gray blocks mark the historical intervals omitted when reconstructing the context. Accuracy is measured against ground-truth subtasks over 15 paired queries per intervention; the displayed predictions are individual examples. For Hidden Object Retrieval, "Random guess" summarizes bin-selection behavior rather than a verbatim prediction.

should determine, indicating that SNaP uses accumulated context for spatial recall, cue interpretation, and progress-dependent decisions.

5 Discussion and Future Work

We introduced SNaP, a self-narrating VLA policy that turns its own subtask descriptions into context for future control. Narration and proprioceptive history are written online into two fixed-size fast-weight banks whose readouts condition subsequent narrations and actions, so the policy builds and uses its context without an expanding text history or a separate narration model. Across three real-robot tasks, SNaP achieves 90.0% average success, outperforming the strongest evaluated baseline by 23.3 percentage points. Targeted interventions further show that subsequent subtask predictions depend on specific earlier information. Together, these results support self-narration as an effective interface for building persistent task context for long-horizon control.

Several limitations remain. First, incorrect narrations written into memory can propagate through the accumulated context; the confidence gate on writes limits but does not remove this effect. Developing mechanisms to verify and revise stored information using subsequent observations is therefore an important direction for future work. Second, training also relies on subtask-annotated demonstrations, motivating approaches that reduce annotation requirements. Finally, our evaluation covers three tasks on a single robot platform with episodes of up to five minutes, so retention over longer and more diverse histories remains untested. Future work should examine how memory capacity and event diversity affect retention, and extend evaluation to broader tasks and embodiments. Replacing the fixed decay with learned, content-dependent forgetting is a natural next step. The same interface could also carry other self-generated signals, such as failure or progress reports. We hope SNaP encourages further work on policies that build and maintain their own task context.

AI USE STATEMENT

(This section is **required** and does not count toward the page limit.)

In this work, we used generative AI tools to propose and refine research ideas, to provide feedback on research methodology and experiment design, and to implement methods, where LLM-based coding assistants helped write, refactor, and debug parts of the training, inference, and evaluation code. We have not used generative AI tools to generate synthetic data sets, develop theoretical models or conceptual frameworks, formulate mathematical claims, clean or reformat datasets, or interpret results, and assisting with the proof of mathematical claims, translation, and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools for brainstorming, identifying and summarizing relevant literature, sourcing information, creating and editing software code, drafting parts of the paper, editing the paper to improve readability, and formatting references. We have reviewed all AI-assisted work. All LLM-assisted code was reviewed and tested by the authors and validated through the real-robot experiments reported in this paper; AI-suggested ideas and methodological feedback were adopted only after the authors' own assessment, and the final method, experiment design, demonstration collection, narration annotation, and interpretation of results are the authors' own; literature identified with AI tools was read and checked against the original sources before being cited; and all AI-drafted or AI-edited text was verified by the authors for factual accuracy and consistency with our method and results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Suneel Belkhale, Tianli Ding, Ted Xiao, Pierre Sermanet, Quan Vuong, Jonathan Tompson, Yevgen Chebotar, Debidatta Dwibedi, and Dorsa Sadigh. RT-H: Action hierarchies using language. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025a. doi: 10.15607/RSS.2025.XXI.010.

Kevin Black, Allen Z. Ren, Michael Equi, and Sergey Levine. Training-time action conditioning for efficient real-time chunking. *arXiv preprint arXiv:2512.05964*, 2025b. URL `https://arxiv.org/abs/2512.05964`.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S. Ryoo, Grecia Salazar, Pannag R. Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan H. Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics transformer for real-world control at scale. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023. doi: 10.15607/RSS.2023.XIX.025.

Aydar Bulatov, Yuri Kuratov, and Mikhail Burtsev. Recurrent memory transformer. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Tianxing Chen, Yuran Wang, Mingleyang Li, Yan Qin, Hao Shi, Zixuan Li, Yifan Hu, Yingsheng Zhang, Kaixuan Wang, Yue Chen, Hongcheng Wang, Renjing Xu, Ruihai Wu, Yao Mu, Yaodong Yang, Hao Dong, and Ping Luo. RMBench: Memory-dependent robotic manipulation benchmark with insights into policy design. *arXiv preprint arXiv:2603.01229*, 2026.

Egor Cherepanov, Nikita Kachaev, Daniil Zelezetsky, Aydar Bulatov, Artem Pshenitsyn, Yuri Kuratov, Alexey Skrynnik, Aleksandr I. Panov, and Alexey K. Kovalev. μVLA: On recurrent memory for partially observable manipulation in VLA models. *arXiv preprint arXiv:2606.12497*, 2026.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Yinpei Dai, Hongze Fu, Jayjun Lee, Yuejiang Liu, Haoran Zhang, Jianing Yang, Chelsea Finn, Nima Fazeli, and Joyce Chai. RoboMME: Benchmarking and understanding memory for robotic generalist policies. In *International Conference on Machine Learning (ICML)*, 2026.

Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *International Conference on Learning Representations (ICLR)*, 2024.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R. Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024. doi: 10.15607/RSS.2024.XX.090.

Sanjay Haresh, Daniel Dijkman, Apratim Bhattacharyya, and Roland Memisevic. Notes-to-self: Scratchpad augmented VLAs for memory dependent manipulation tasks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1769–1782. PMLR, 2023.

Huiwon Jang, Sihyun Yu, Heeseung Kwon, Hojin Jeon, Younggyo Seo, and Jinwoo Shin. ContextVLA: Vision-language-action model with amortized multi-frame context. *arXiv preprint arXiv:2510.04246*, 2025.

Yunfan Jiang, Yevgen Chebotar, Ruijie Zheng, Fengyuan Hu, Yunhao Ge, Jimmy Wu, Tianyuan Dai, Scott Reed, Fei-Fei Li, Yuke Zhu, and Linxi "Jim" Fan. RoboTTT: Context scaling for robot policies. *arXiv preprint arXiv:2607.15275*, 2026. URL https://arxiv.org/abs/2607.15275.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. URL https://arxiv.org/abs/2406.09246.

Myungkyu Koo, Daewon Choi, Taeyoung Kim, Kyungmin Lee, Changyeon Kim, Younggyo Seo, and Jinwoo Shin. HAMLET: Switch your vision-language-action model into a history-aware policy. In *International Conference on Learning Representations (ICLR)*, 2026.

Huashuo Lei, Wenxuan Song, Huarui Zhang, Jieyuan Pei, Jiayi Chen, Haodong Yan, Han Zhao, Pengxiang Ding, Zhipeng Zhang, Lida Huang, Donglin Wang, Yan Wang, and Haoang Li. RoboMemArena: A comprehensive and challenging robotic memory benchmark. *arXiv preprint arXiv:2605.10921*, 2026.

Hang Li, Fengyi Shen, Dong Chen, Liudi Yang, Xudong Wang, Jinkui Shi, Zhenshan Bing, Ziyuan Liu, and Alois Knoll. ReMem-VLA: Empowering vision-language-action model with memory via dual-level recurrent queries. *arXiv preprint arXiv:2603.12942*, 2026a.

Wenzhuo Li, Qiongfeng Shi, and Yi Zhou. StateMem: Single-state residual memory with adaptive inference for vision-language-action policies. *arXiv preprint arXiv:2609.22684*, 2026b. URL https://arxiv.org/abs/2609.22684.

Max Sobol Mark, Jacky Liang, Maria Attarian, Chuyuan Fu, Debidatta Dwibedi, Aviral Kumar, and Dhruv Shah. BPP: Long-context robot imitation learning by focusing on key history frames. *arXiv preprint arXiv:2602.15010*, 2026.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*, 2024.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025. URL https://arxiv.org/abs/2504.16054.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations (ICLR)*, 2020.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9355–9366. PMLR, 2021. URL https://proceedings.mlr.press/v139/schlag21a.html.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. MemoryVLA: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/1fa3d6ccbcd15f7285fee666b2bd57be-Abstract-Conference.html.

Lucy Xiaoyang Shi, Brian Ichter, Michael Robert Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. Hi Robot: Open-ended instruction following with hierarchical vision-language-action models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 54919–54933. PMLR, 2025.

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. Scaling up memory for robotic control via experience retrieval. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/9da515b1ad19d032a7398f00f5ff9b0c-Abstract-Conference.html.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): RNNs with expressive hidden states. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 57503–57522. PMLR, 2025.

Marcel Torne, Andy Tang, Yuejiang Liu, and Chelsea Finn. Learning long-context diffusion policies via past-token prediction. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 1744–1755. PMLR, 2025.

Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z. Ren, Haohuan Wang, Jiaming Tang, Kyle Stachowicz, Karan Dhabalia, Michael Equi, Quan Vuong, Jost Tobias Springenberg, Sergey Levine, Chelsea Finn, and Danny Driess. MEM: Multi-scale embodied memory for vision language action models. *arXiv preprint arXiv:2603.03596*, 2026. URL https://arxiv.org/abs/2603.03596.

Chuan Wen, Jierui Lin, Trevor Darrell, Dinesh Jayaraman, and Yang Gao. Fighting copycat agents in behavioral cloning from observation histories. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. GELLO: A general, low-cost, and intuitive teleoperation framework for robot manipulators. *arXiv preprint arXiv:2309.13037*, 2023. URL https://arxiv.org/abs/2309.13037.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Cheng Yin, Wang Xu, Junpeng Yang, Sikyuen Tam, Hanyu Liu, Yuan Yao, Xiangrui Zeng, Junbo Cui, Yequan Wang, Zhouping Yin, and Yankai Lin. SimpleMemVLA: A simple but effective native-video memory for vision-language-action models. *arXiv preprint arXiv:2609.05533*, 2026. URL https://arxiv.org/abs/2609.05533.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 3157–3181. PMLR, 2025.

A Implementation Details

A.1 Fast-Weight Context Banks

Narration bank. For an accepted narration $y_t = (x_{t,0}, \ldots, x_{t,L_t-1})$, the narration bank writes one key–value pair for each valid narration token. We re-encode the narration using a causal, text-only pass that shares the policy's token embedder and early Transformer blocks. The resulting backbone features are detached before entering the write path. For token $x_{t,j}$, the key represents its causal textual context, whereas the value represents the token itself:

$$k^N_{t,j} = f^N_K\big(\mathrm{sg}(h^{\mathrm{ctx}}_{t,j})\big)\,, \qquad v^N_{t,j} = f^N_V\big(\mathrm{sg}(E(x_{t,j}))\big)\,. \tag{10}$$

Here, E is the shared token embedder and sg denotes stop-gradient. For noninitial positions, $h^{\mathrm{ctx}}_{t,j}$ is obtained from the preceding causal hidden state. The initial position uses the first available causal state; the first two token positions therefore share the same initial contextual state before the key projection.

Context features and token embeddings are standardized using statistics computed from a fixed task-specific narration reference set whose sentences occur in the training split. Projected contextual keys are further transformed using a regularized PCA-based whitening map and then ℓ_2 normalized. These reference statistics are used only for feature transformation and do not restrict generated narrations to a fixed sentence template.

Sensory bank. The sensory bank retains the robot's proprioceptive history. The physical state contains 14 channels, corresponding to six joints and one gripper value for each arm. After the same normalization used by the policy input pipeline, the state is padded to the model's 32-dimensional state interface. A learned state encoder first computes

$$u_t = f_{\mathrm{state}}(\mathrm{sg}(p_t))\,, \tag{11}$$

followed by separate key and value encoders:

$$k^S_t = f^S_K(u_t), \qquad v^S_t = f^S_V(u_t). \tag{12}$$

The sensory bank writes one state-dependent key–value pair at every valid decision tick, independently of whether the narration bank commits a new narration.

A.2 Fast-Weight Update

Both banks use a 512-dimensional key/query space and a 2048-dimensional value space. The nonlinear SNaP configuration uses an address network with three hidden layers of width 1024. Each hidden layer uses a SiLU activation followed by feature normalization. Only the final fast-weight matrix is modified within an episode; the address-network parameters and all other slow model parameters remain fixed during deployment.

At each valid decision tick, retention is applied once before processing the available key–value pairs. We use retention factor $\rho = 0.99$ and update strength $\eta = 1$. Given a pair (k_j, v_j), the address network produces $z_j = \phi_\theta(k_j)$ and the current residual is

$$\delta_j = v_j - \bar{W}_j^\top z_j. \tag{13}$$

The pair is incorporated through

$$\bar{W}_{j+1} = \bar{W}_j + c\,\eta\,\frac{z_j \delta_j^\top}{\|z_j\|_2^2}, \qquad \bar{W}_0 = \rho W, \tag{14}$$

where $c \in \{0, 1\}$ is the bank's write gate: $c = 1$ for every sensory write and for the token pairs of an accepted narration ($c = c^N_t$ in Algorithm 1), and $c = 0$ for a rejected narration, which then receives retention only. Pairs are processed sequentially, so each residual is computed using the matrix produced by earlier writes at the same decision tick.

For model-generated narration writes, a single sentence-level decision gates all valid narration-token pairs. A narration is eligible for writing when it is nonempty, differs from the most recently committed narration, and its mean selected-token probability is at least 0.9. Rejected narrations undergo retention without a content update. Invalid or padded ticks leave the bank unchanged. The sensory bank instead writes at every valid decision tick.

A.3 Context Readout and Integration

Each active bank uses $K = 8$ learned read queries. The query generator conditions learned query bases on the current early multimodal prefix representations and on the most recently committed narration. Each query then accesses the corresponding address network and incoming fast-weight matrix:

$$q^b_{t,i} = \mathcal{Q}_b\big(H_t, \bar{y}_{t-1}; B^b_i\big), \qquad r^b_{t,i} = (W^b_{t-1})^\top \phi^b_{\theta_b}(q^b_{t,i}). \tag{15}$$

For the default narration-plus-sensory configuration, the two banks therefore produce 16 active readouts. Read queries are constructed after the first nine Transformer blocks, corresponding to zero-based block index 8. The retrieved vectors are inserted at this intermediate boundary and are available to subsequent Transformer blocks.

Before insertion, each raw readout is scale-calibrated and gated:

$$m^b_{t,i} = \mathrm{cast}\left[g^b r^b_{t,i} \frac{12.4}{\max(\mathrm{RMS}(r^b_{t,i}), 0.02)} + e^b_i\right], \tag{16}$$

where $\mathrm{RMS}(\cdot)$ is the root mean square over feature dimensions. The constant 12.4 is the target RMS of an inserted context token, chosen to match the typical RMS of the backbone's prefix hidden states at the insertion layer, and the floor of 0.02 prevents amplifying near-zero readouts from an empty bank. The gain $g^b = 0.5$ in the evaluated configuration, e^b_i is a learned read-position embedding, and cast converts the result to the backbone's activation precision. The resulting context tokens can be attended to by the subsequent narration and action paths.

Importantly, the current narration also conditions the action expert directly at the same decision tick. It does not first need to be written to the narration bank and retrieved again. Writes performed after the current prediction affect subsequent decisions.

A.4 Online Context Update

Algorithm 1 summarizes the causal ordering of context retrieval, narration generation, action prediction, and fast-weight updates. $\mathrm{ReadBank}$ implements the K-query readout of Eq. 15, and Decode_θ returns the greedily decoded narration together with the probabilities $s_{t,j}$ of its selected tokens, which the commit rule averages. $\mathrm{WriteBank}(W, \mathcal{P}, c)$ applies retention to W and then the gated update of Eq. 14 to the key–value pairs $\mathcal{P}$ with write gate c; $\mathrm{EncodeSensory}$ and $\mathrm{EncodeNarration}$ produce these pairs via Eqs. 12 and 10.

B Training Details

B.1 Two-Stage Sequence Training

SNaP is initialized from a narration-conditioned $\pi_{0.5}$ policy and trained on chronological demonstration windows. Training proceeds in two stages.

In the first stage, annotated narrations provide oracle writes to the narration bank. In the second stage, narration-bank writes use model predictions obtained by taking the argmax of teacher-forced narration logits. The transition between these stages is a discrete switch rather than a gradual mixture of oracle and predicted writes.

The predicted write candidate used in the second stage is distinct from deployment-time narration generation. During training, the candidate is constructed under the ground-truth narration prefix and is then re-encoded to obtain narration-bank key–value pairs. The action branch remains conditioned on the annotated current narration. During deployment, narration is instead generated autoregressively and directly conditions the current action.

B.2 Training Objective

We optimize SNaP using continuous-action flow matching together with cross-entropy over narration and FAST action tokens (Pertsch et al., 2025):

$$\mathcal{L} = \mathcal{L}_{\mathrm{flow}} + \lambda_{\mathrm{CE}} \mathcal{L}_{\mathrm{token}}. \tag{17}$$

Algorithm 1 SNaP online context read–write loop

Require: Task instruction ℓ and observation sequence $o_t = (I_t, p_t)$

Initialize $W_0^N = W_0^S = 0$
Initialize $\bar{y}_0 = \varnothing$
for decision tick $t = 1, 2, \ldots$ **do**
  $H_t \leftarrow \mathrm{EarlyVLA}_\theta(o_t, \ell)$
  $R_t^N \leftarrow \mathrm{ReadBank}(W_{t-1}^N, H_t, \bar{y}_{t-1})$
  $R_t^S \leftarrow \mathrm{ReadBank}(W_{t-1}^S, H_t, \bar{y}_{t-1})$
  $R_t \leftarrow [R_t^N; R_t^S]$
  $y_t, \{s_{t,j}\} \leftarrow \mathrm{Decode}_\theta(o_t, \ell, R_t)$
  $A_t \sim \pi_\theta^{\mathrm{act}}(\cdot \mid o_t, \ell, R_t, y_t)$
  $(k_t^S, v_t^S) \leftarrow \mathrm{EncodeSensory}(p_t)$
  $W_t^S \leftarrow \mathrm{WriteBank}(W_{t-1}^S, \{(k_t^S, v_t^S)\}, 1)$
  $c_t^N \leftarrow \mathrm{Valid}(y_t) \wedge (y_t \neq \bar{y}_{t-1}) \wedge \left(\frac{1}{L_t}\sum_j s_{t,j} \geq 0.9\right)$
  if c_t^N **then**
    $\mathcal{P}_t^N \leftarrow \mathrm{EncodeNarration}(y_t)$
    $W_t^N \leftarrow \mathrm{WriteBank}(W_{t-1}^N, \mathcal{P}_t^N, 1)$
    $\bar{y}_t \leftarrow y_t$
  else
    $W_t^N \leftarrow \mathrm{WriteBank}(W_{t-1}^N, \varnothing, 0)$
    $\bar{y}_t \leftarrow \bar{y}_{t-1}$
  end if
end for

The token loss is computed over the shared causal sequence containing both narration and FAST action tokens. No separate reconstruction objective is used for the narration or sensory fast-weight banks in the final SNaP training objective.

For the action objective, the policy predicts a horizon of 50 control actions. Training-time real-time chunking (RTC) samples a clean action prefix of up to 15 control steps and applies the flow-matching objective to the remaining suffix. The action branch is conditioned on the current observation, retrieved context, and annotated current narration during training.

B.3 Context Prefill and Sequence Windows

Training uses short chronological windows rather than unrolling complete long-horizon demonstrations through the VLA. At the beginning of each window, the two fast-weight states are reconstructed from bounded preceding history.

For the narration bank, earlier annotated narration changes are replayed in temporal order. Retention between narration writes is preserved during this replay. For the sensory bank, preceding proprioceptive observations are replayed using the same state encoder and fast-weight update used in the current window.

We use truncated backpropagation through time (TBPTT) within sampled sequence windows to limit activation storage while carrying the fast-weight states forward. Independent sampled windows do not inherit fast-weight states from unrelated examples; each window reconstructs its own preceding context before processing the current sequence.

B.4 Optimization and Deployment

Memory-stage optimization uses AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and gradient clipping. The current visual encoder is frozen during memory-stage training, while the VLA language blocks, action expert, context encoders, query generators, and slow address-network parameters are optimized through the outer training loop. The fast-weight matrices themselves are episode state rather than ordinary optimizer parameters.

At deployment, each episode starts with zero fast-weight matrices. Narration is generated greedily and autoregressively. The policy reads the incoming banks, generates the current narration, predicts an action chunk, and only then applies the sensory and narration writes shown in Algorithm 1. All slow model parameters remain fixed during deployment.

C Experimental Details

C.1 Robot and Observation Setup

All experiments use the dual-arm YAM platform described in the main text. Each arm has six actuated joints and one gripper degree of freedom. Demonstrations are collected through GELLO teleoperation (Wu et al., 2023).

The policy receives RGB observations from one third-person camera and two wrist-mounted cameras. Each image is resized to 224×224 using aspect-ratio-preserving padding. The robot operates at 30 Hz, and each policy inference predicts a 50-step action chunk. We use training-time RTC (Black et al., 2025b) with a maximum delay budget of 15 control steps.

The three tasks are trained separately rather than through a single three-task checkpoint. Task-state narrations are aligned with the demonstration trajectories and provide supervision for self-narration training.

C.2 Task-Specific Context

Hidden Object Retrieval. The policy must retain an object–location relation observed before the bins are closed. The language instruction identifies the target object but does not specify its bin. The later retrieval decision therefore depends on previously observed spatial information.

Cued Bean Scooping. The number of green flashes specifies a target of one to three scoops. The target number is not stated explicitly in the language instruction. The policy must retain the earlier cue while tracking the number of completed scoop-and-pour cycles.

Two-Drink Recipe. The evaluated recipe requires three scoops of boba for the first drink and one scoop of red beans for the second. The instruction specifies the drink order, while the recipe counts are represented in the demonstrations and task-state narration supervision. The task therefore combines ingredient location recall with count and progress tracking.

C.3 Context-Modality Ablations

All variants in Table 1 include the token-level narration bank. The Linear configuration removes the three hidden layers from the address network, whereas the Nonlinear configuration uses the three-layer address network employed by SNaP.

The Visual and Sensory columns indicate optional sources of historical context. They do not remove the current visual observations or current proprioceptive state from the VLA input.

Sensory context contributes one proprioceptive key–value pair at each valid decision tick. Visual context is constructed from pooled features of the current front-camera representation and contributes multiple visual key–value pairs. When both optional modalities are enabled, their writes share a single auxiliary fast-weight bank rather than forming two additional independent banks.

The default SNaP configuration used for the main comparison consists of the nonlinear narration bank together with the nonlinear sensory bank, without historical visual writes.

C.4 Evaluation Protocol

For Q1 and Q2, every reported task success rate is evaluated over 30 real-robot trials per task. Success is evaluated at the task level: the complete instructed behavior must be executed correctly for the trial to count as successful.

For Q3, we instead evaluate subtask prediction under paired context interventions. Each intervention contains 15 paired queries, as described below.

D CONTEXT INTERVENTION PROTOCOL

The context interventions in Q3 test whether a later subtask prediction depends on a selected earlier event. For each pair, we use the same trained SNaP policy, current query observation, task instruction, and model parameters. The only difference is the historical information used to reconstruct the context available before the query.

We compare:

1. **Full context**, reconstructed from the complete relevant history before the query; and
2. **Event-removed context**, reconstructed while omitting the selected earlier task event.

The current query observation itself is unchanged. Each prediction is compared against the same ground-truth subtask, so the intervention measures subtask-prediction accuracy rather than full closed-loop task success.

Hidden Object Retrieval. We remove the earlier evidence establishing the banana location and query the policy after the bins have been closed. With full context, the policy predicts the correct target location in $15/15$ paired queries. With the corresponding event removed, it is correct in $4/15$ queries.

Cued Bean Scooping. We remove one green blink from the earlier cue sequence and query the policy at the subsequent yellow "go" decision. Full-context predictions are correct in $14/15$ queries, whereas the event-removed predictions are correct in $0/15$.

Two-Drink Recipe. We remove one completed boba-scoop event from the earlier context and query the policy at the subsequent progress-dependent decision. The full-context prediction is correct in $14/15$ queries, whereas the event-removed prediction is correct in $0/15$.

Figure 5 displays one representative example for each intervention. The displayed predictions illustrate individual cases, whereas the accuracy values summarize all 15 paired queries. In the Hidden Object example, "random guess" summarizes the resulting bin-selection behavior rather than a verbatim decoded narration.