LATCH: ACTION-GUIDED PERCEPTUAL MEMORY FOR VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Non-Markovian manipulation requires a vision-language-action (VLA) policy to know which past observations still matter for the action being produced. Most VLAs act from the current observation or a short visual window; existing memory systems either add symbolic task state through a separate high-level policy trained on subtask or keyframe annotation, or retain perceptual history with no task-general way to prioritize useful past observations. We introduce LATCH, which learns this priority inside the policy: lightweight read and write gates control where the action queries attend within a uniformly sampled visual history and how strongly each frame contributes, optimized jointly with the VLA action objective. The action objective is the only supervision signal: no annotation pipeline, no retrieval policy, and a single differentiable policy at inference. We evaluate LATCH on the $\pi_{0.5}$ and GR00T N1.6 backbones across RoboMME's 16 memory-centric tasks, the LIBERO suites, dual-arm RMBench, and real-world memory-centric tasks on an SO-101 arm. On RoboMME, LATCH lifts $\pi_{0.5}$ from 17.9% to 47.2% average success, higher than all other evaluated memory methods for VLAs. These results indicate that a bounded perceptual memory can be organized by the action objective alone and can effectively provide historical context without keyframe or subtask annotation and without an external memory policy. Project page: `https://latch-robomem.github.io`.

1 INTRODUCTION

Generalist vision-language-action (VLA) models such as $\pi_{0.5}$ (Physical Intelligence et al., 2025) and GR00T N1 (NVIDIA et al., 2025) execute diverse manipulation tasks with a shared policy. Most, however, remain largely Markovian: they predict actions from the current observation or a short visual window (Koo et al., 2026; Dai et al., 2026). This formulation breaks down when the current scene does not reveal the task state: a robot may need to remember how many objects it has moved, where an object was placed before it became occluded, or which motion it was shown earlier, so the same observation can require different actions depending on the preceding trajectory. Memory is therefore part of the state representation for non-Markovian manipulation, not simply additional visual context.

Long visual context is costly. Language models operate over ever longer token sequences (Ding et al., 2024), but robot trajectories contain dense, high-dimensional images that must be processed under control-time latency, and extending visual history indiscriminately adds computation, redundancy, and interference (Torne et al., 2025; Koo et al., 2026; Sridhar et al., 2026). The useful history is also sparse: a brief interaction or state change may determine an action many steps later. An effective robot memory must therefore cover long horizons while selecting a small amount of decision-relevant evidence.

Long-horizon VLA methods differ in what they remember and how it reaches control. Symbolic approaches use a high-level policy to express history as task state or subgoals; perceptual methods retain visual evidence within the action policy, as a compressed context token (Jang et al., 2025), time-contrastive moment tokens (Koo et al., 2026), or a perceptual-cognitive memory bank (Shi et al., 2026); hybrids combine both, MemER (Sridhar et al., 2026) retrieving keyframes while a high-level VLM predicts the current subtask, and MEM (Torne et al., 2026) coupling compressed video with a language summary of progress. The high-level compo-

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.

nents of MemER and MEM are trained on subtask-annotated trajectories; perceptual methods avoid that interface, but task-general selection of visual evidence remains unresolved. RoboMME (Dai et al., 2026) makes the gap concrete: across 14 memory variants on one $\pi_{0.5}$ backbone, no representation is best in every task family, and the strongest, MME-VLA, modulates the action expert with 32 historical frames that are still chosen uniformly in time. The remaining question is how to learn this priority from signals already available in robot demonstrations.

We introduce *LATCH*, an action-guided perceptual memory mechanism for VLA models. LATCH begins with a uniformly sampled visual history and learns two lightweight gates: a read gate that biases where the action queries attend within the history, and a write gate that scales how strongly each frame contributes. Both are optimized with the VLA action objective alone, so memory priority is learned from control rather than from a separate retrieval policy, and LATCH remains a single differentiable policy that needs neither keyframe or subtask labels nor an external high-level memory policy.

We evaluate LATCH on $\pi_{0.5}$ and GR00T N1.6 (NVIDIA et al., 2025) across the memory-centric RoboMME (Dai et al., 2026) and RMBench (Chen et al., 2026b) benchmarks, the general-manipulation LIBERO (Liu et al., 2023) suites, and three real-world memory tasks on an SO-101 arm (Cadene et al., 2024). On RoboMME, LATCH reaches 47.2% average success on $\pi_{0.5}$, the highest among the evaluated memory methods, ahead of the uniform memory it builds on and of the annotated hybrid, and the same ordering holds on GR00T N1.6. The mechanism carries over unchanged to a second backbone, a dual-arm embodiment, and a physical arm: it is the strongest perceptual memory on RMBench and the most reliable at keeping a count on the real robot, while costing nothing on regular tasks. Controlled ablations then isolate what the result rests on and what each gate contributes. Our contributions are:

- *LATCH*, an action-guided perceptual memory: read and write gates that learn, from the action objective alone, which frames of a uniformly sampled history influence each action, inside a pretrained VLA, without annotation or an external memory policy.
- The highest RoboMME average among the evaluated memory methods on both $\pi_{0.5}$ and GR00T N1.6, ahead of the uniform memory it builds on and of the annotated hybrid.
- Transfer beyond the primary benchmark: the strongest perceptual memory on the dual-arm RMBench, matching the annotated hybrid overall, the most reliable counting on a real SO-101 arm, and no cost on LIBERO's regular tasks.

2 Related Work

Vision-language-action models. Generalist policies such as RT-2 (Brohan et al., 2023), OpenVLA (Kim et al., 2024), $\pi_{0.5}$ (Physical Intelligence et al., 2025), and GR00T N1 (NVIDIA et al., 2025) transfer vision-language representations to robotic control through action tokenization or generative action heads. These models remain largely Markovian: each action prediction depends on the current observation or a short recent window, so identical current observations cannot be distinguished by events that occurred earlier in the episode. Hierarchical systems such as Hi Robot (Shi et al., 2025) and HAMSTER (Li et al., 2025) add a high-level VLM that issues language subgoals or coarse trajectories, but the executor still maps the current observation and command to actions. History-conditioned methods relax this assumption, Past-Token Prediction (Torne et al., 2025) by training on extended observation histories and ContextVLA (Jang et al., 2025) by compressing frames into a context token, but leave open how a bounded visual context should prioritize evidence for the current decision.

Perceptual memory. Long visual context in VLAs is not simply a longer context window: a robot acts in a closed loop, so its predictions change the observations that enter its future context, and behavior cloning can exploit historical correlations, such as cues to previous expert actions, that fail after policy deviation (de Haan et al., 2019; Wen et al., 2020). Past-Token Prediction (Torne et al., 2025) identifies a related mismatch in diffusion policies: additional history does not by itself preserve expert temporal dependencies. Visual history must also fit a finite memory and latency budget: MemER (Sridhar et al., 2026) reports that a 32-frame context covers about 16 seconds, approaches one second of inference latency, and still underperforms selected keyframe

memory. Perceptual memory must therefore prioritize task-relevant evidence within a bounded visual context.

HAMLET (Koo et al., 2026) aggregates moment tokens, MemoryVLA (Shi et al., 2026) retrieves from a perceptual-cognitive memory bank, SAM2Act+ (Fang et al., 2025) stores spatial features, and HistRISE (Chen et al., 2026a) compresses history into object-centric point tracks. RoboMME (Dai et al., 2026) compares such variants under a common $\pi_{0.5}$ backbone: FrameSamp retains uniformly sampled frames and TokenDrop selects patches by local visual change, entering the policy as context, modulator, or dedicated expert. FrameSamp+Modul is the strongest baseline in that comparison, but its memory is still selected uniformly in time, and TokenDrop responds to visual change without learning action relevance. LATCH gives visual history decision-dependent influence through action-guided gates, learning memory priority from signals already present in demonstrations.

Symbolic, semantic, and hybrid memory. Symbolic and semantic memory compress interaction history into explicit task-level state rather than retaining raw observations: UniVLA (Bu et al., 2025) feeds its previous latent action back as prompt tokens, and Gemini Robotics 1.5 (Gemini Robotics Team et al., 2025) pairs a VLM orchestrator with a VLA to maintain task progress and issue executable instructions. These representations are compact and interpretable, but their fixed interfaces or separate reasoning stages can omit the spatial and motion details needed for low-level control. Hybrid methods retain perceptual evidence alongside semantic state: MEM (Torne et al., 2026) combines compressed short-term video with a recursively updated language summary whose targets are generated from subtask-annotated episodes by an external LLM, and MemER (Sridhar et al., 2026) retrieves visual keyframes and predicts the current subtask with a high-level policy trained on subtask labels and rule-based keyframe targets. LATCH keeps memory within the differentiable action policy, without subtask or keyframe annotation.

Memory-centric evaluation. General manipulation benchmarks such as LIBERO (Liu et al., 2023), CALVIN (Mees et al., 2022), and RoboCasa (Nasiriany et al., 2024) evaluate task transfer, long-horizon instruction following, or household manipulation, but do not isolate memory as the principal variable. MemoryBench (Fang et al., 2025) probes spatial recall and action history in a few tasks, MIKASA-Robo (Cherepanov et al., 2026) organizes 32 tabletop tasks around a broad memory taxonomy, and RMBench (Chen et al., 2026b) stratifies nine dual-arm tasks by the number of past observations an optimal decision requires. RoboMME (Dai et al., 2026) evaluates 16 non-Markovian tasks spanning counting, object permanence, reference, and imitation while controlling the backbone and memory budget across memory designs, and is our primary benchmark, with LIBERO (Liu et al., 2023) as the complementary general-manipulation evaluation.

3 METHOD

LATCH endows a vision-language-action policy with a perceptual memory whose readout is learned from the action objective alone. Its starting point is an observation: whether a past frame matters is not a property of the frame itself but of the frame relative to the action currently being produced. An attention-based memory readout responds to exactly two levers—where its queries look, and how loudly the attended content is aggregated—and LATCH makes both levers learned, per-frame functions. Two small gates inside the memory cross-attention, a *read gate* on the attention logits and a *write gate* on the value vectors, are optimized end to end through control. Training uses no keyframe or subtask annotation and no auxiliary loss, and inference needs no retrieval policy. We instantiate LATCH on the $\pi_{0.5}$ (Physical Intelligence et al., 2025) and GR00T N1.6 (NVIDIA et al., 2025) backbones; Figure 1 summarizes the architecture.

Problem formulation. At control step t, the policy receives a language instruction ℓ, the current observation o_t, and the robot state s_t, and predicts a chunk of H future actions. Memory-dependent manipulation additionally requires evidence from the observation history $\mathcal{H}_t = \{o_1, \ldots, o_{t-1}\}$, whose task state the current frame does not reveal. Carrying $\mathcal{H}_t$ at full visual resolution is infeasible under control-time latency, so the policy maintains a bounded perceptual memory M_t of at most B visual tokens.

Memory-conditioned policy. The memory follows the uniform-sampling protocol of MME-VLA (Dai et al., 2026). At each step, N frames are sampled uniformly in time from the available

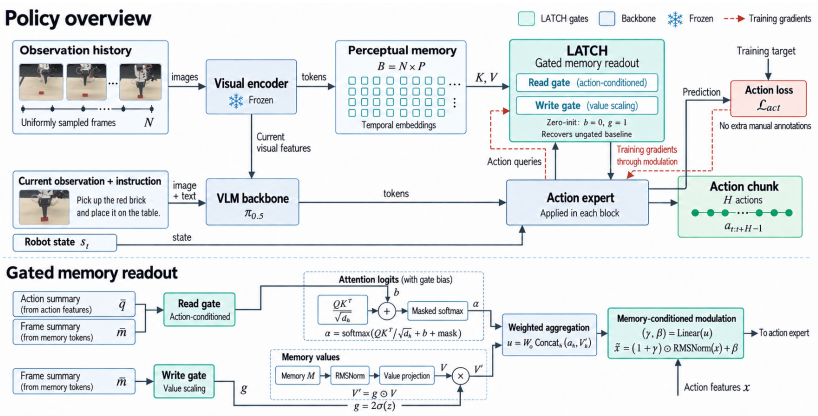


Figure 1: LATCH overview. A bounded perceptual memory of N frames sampled uniformly from the observation history, encoded to P tokens each, conditions the action expert through cross-attention with feature-wise modulation. LATCH adds two per-frame gates to the readout: a query-aware *read gate* biasing the attention logits (where the action queries look) and a query-blind *write gate* scaling each frame's values after normalization (how loudly its content is aggregated). Both gates are zero-initialized, and all memory priority is learned through the action objective alone.

history and encoded by the frozen vision encoder into P tokens each; learned temporal position embeddings are added, and the tokens are projected to the action-expert width d, yielding

$$M_t = [\, m_1; \ldots; m_N \,], \qquad m_i \in \mathbb{R}^{P \times d}, \tag{1}$$

with budget $B = NP$ tokens. Memory conditions the policy through feature-wise modulation (Perez et al., 2018) rather than context concatenation. In every block of the action expert, the normalized action hidden states n_t produce queries that attend over the memory,

$$u_t = \textstyle\sum_{i,p} \alpha_{t,ip}\, V_{i,p}, \qquad \alpha_{t,ip} = \mathrm{softmax}_{ip}\big(q_t \cdot k_{i,p}/\sqrt{d}\big), \tag{2}$$

and the readout u_t is mapped to a per-token scale and shift that modulates the block,

$$\widetilde{x}_t = n_t \odot \big(1 + \gamma(u_t)\big) + \beta(u_t), \tag{3}$$

where (γ, β) is a learned projection, in the modulation form of MME-VLA (Dai et al., 2026). In this policy nothing in the readout distinguishes one moment of the history as more relevant than another.

Gated memory readout. LATCH generalizes the readout in (2)–(3) by turning its two implicit constants into learned functions of the memory and the action state. Let $\bar{m}_i$ denote the mean of frame i's normalized memory tokens and $\bar{a}$ the mean of the normalized action hidden states.

The *write gate* controls how loudly a frame speaks. A two-layer perceptron with GELU activation (Hendrycks & Gimpel, 2016) consumes the frame summary alone,

$$z_i = W_2\, \mathrm{GELU}(W_1 \bar{m}_i), \qquad g_i = 2\,\sigma(z_i) \in [0, 2], \tag{4}$$

and the scale multiplies the frame's value vectors after normalization, $\widetilde{V}_{i,p} = g_i\, V_{i,p}$. The placement is what makes the gate effective: a multiplicative scale inserted before RMS normalization (Zhang & Sennrich, 2019) is divided back out by the normalization itself, whereas the post-normalization scale survives into the readout. The keys are left untouched, so the write gate changes how much a frame contributes without changing where attention is routed.

The *read gate* controls where the action queries look. A second perceptron consumes the frame summary together with the action summary and their element-wise product,

$$b_i = W_4\, \mathrm{GELU}\big(W_3\, [\, \bar{a};\, \bar{m}_i;\, \bar{a} \odot \bar{m}_i \,]\big), \tag{5}$$

and the attention weights in (2) become $\alpha_{t,ip} \propto \exp\big(q_t \cdot k_{i,p}/\sqrt{d} + b_i\big)$—one additive scalar per frame, shared across heads and query positions. Because the bias is additive it is immune to normalization, and because it is computed from $\bar{a}$ it is query-aware.

The two gates are complementary in what they act on and asymmetric in what they see: the read gate biases the logits that route attention, the write gate scales the values that are aggregated; the

write gate is *query-blind*, so frame content alone sets how loudly a frame may speak, whereas the read gate is *query-aware* and weighs the same frame differently as the action state changes. Both output layers are zero-initialized, so $b_i{=}0$ and $g_i{=}1$ for every frame: the gated policy starts exactly as the memory-conditioned policy above and departs from equal weighting only where the action objective demands it. The gates add orders of magnitude fewer parameters than the policy and sit in the gradient path from action prediction to memory, so memory priority is learned from control rather than imposed by a retrieval label.

Training protocol. LATCH trains in two phases under the same action objective. In the first phase, the memory-conditioned policy above is trained with the action objective; in the second, the gates are attached with their zero initialization, recovering the phase-one policy exactly at step zero, and the full policy is finetuned end to end with the action loss alone, $\mathcal{L} = \mathcal{L}_{\text{act}}$. At inference, LATCH consumes the same inputs as the phase-one policy—instruction, current observation, robot state, and the sampled history—and the gates amount to two perceptron evaluations per frame, leaving the runtime profile of the policy unchanged. The construction presumes only that the policy reads memory through attention, which holds across the VLA backbones we consider. Instantiation constants, training schedules, and per-dataset protocols are given in Appendix A.

4 Experiments

Our experiments ask whether action-guided perceptual memory is a general mechanism rather than a benchmark-specific one: we compare LATCH with matched perceptual memory baselines and with annotation-trained hybrids on the primary memory benchmark, then across a second VLA architecture, a dual-arm memory benchmark, regular tasks on which memory must not cost anything, and a real robot, and isolate its design choices with a controlled ablation. Section 4.1 describes the setup; Section 4.2 reports and discusses the results question by question.

4.1 Experiment Setup

Datasets and benchmarks. *RoboMME* (Dai et al., 2026) is the primary benchmark: 16 non-Markovian tabletop tasks (Franka Panda, ManiSkill (Mu et al., 2021)), four per family (Counting, Permanence, Reference, Imitation), with 100 demonstrations each. Each family withholds a different kind of state from the current frame, so per-family success separates memory representations rather than only ranking their averages. *LIBERO* (Liu et al., 2023) is the well-established control: the Spatial, Object, Goal, and Long suites (ten tasks each, 50 demonstrations, same arm) are largely Markovian and near saturation, and test whether memory costs anything on regular tasks. *RMBench* (Chen et al., 2026b) changes the embodiment to the dual-arm Aloha-AgileX platform of RoboTwin 2.0 (Chen et al., 2025): nine tasks with 50 demonstrations each, five at memory complexity M(1), whose action depends on one past observation, and four at M(n), whose action depends on several accumulated over repeated attempts. *Real-world tasks* use an SO-101 arm (Cadene et al., 2024) with front and wrist cameras on three memory tasks, with a total of 200 demonstrations. The details are provided in Appendix A.4.

Baselines. $\pi_{0.5}$ (Physical Intelligence et al., 2025) and *GR00T N1.6* (NVIDIA et al., 2025) are the memory-free backbones and act from the current observation. As matched baselines we select two recent perceptual memory methods and train them on each backbone with the same demonstrations: *MME-VLA* (Dai et al., 2026), the best-performing memory model in the RoboMME study, which conditions the action expert on 32 uniformly sampled frames through modulation, and *HAMLET* (Koo et al., 2026), which compresses the recent history into moment tokens with a learned memory module; HAMLET does not report RoboMME, so on $\pi_{0.5}$ we train it with its released recipe, and on GR00T N1.6 we evaluate a public RoboMME checkpoint trained with the authors' recipe. On RoboMME we further compare against the published results of *MemER* (Sridhar et al., 2026), a hybrid that retrieves keyframes and predicts language subtasks with a VLM trained on subtask annotation and executes them with a $\pi_{0.5}$ policy, of *MemoryVLA* (Shi et al., 2026) and *SAM2Act+* (Fang et al., 2025), two perceptual memories on their own architectures, and of $\pi_{0.5}$ with an action history. On RMBench the compared methods share the $\pi_{0.5}$ backbone, and we further compare against the benchmark authors' *Mem-0* (Chen et al., 2026b), a hybrid that plans language subtasks over a key memory of completed subtasks with a VLM trained

on subtask annotation and executes them with a diffusion policy using anchor and sliding memories on its own architecture. The tables mark each method's memory type and whether its training needs manual annotation. Appendix A.2 details the selection and training of the baselines.

Training. On the RoboMME benchmark, each baseline keeps the configuration of its published recipe: RoboMME study on $\pi_{0.5}$ and HAMLET on GR00T N1.6. LATCH follows the two phases specified in Section 3. The gate phase is the last $\approx$30–40% of the training budget, and the uniform-memory control is trained for the full budget, so the two are matched in compute. LIBERO, RMBench, and the real robot use one schedule per benchmark, with the same gate-phase split in simulation; on the real robot there is no uniform-memory stage, and LATCH is trained from the base weights. Exact step counts, schedules, and optimizer settings are listed in Appendix A.

Evaluation. The metric is task success rate, averaged per family and over all tasks; every result of ours is the mean $\pm$ standard deviation over three evaluation seeds, and reported values (†) are those of the source paper. *Simulated benchmarks:* On RoboMME we use the full protocol, all 16 tasks with 50 episodes each; LIBERO is likewise evaluated on its full protocol, 50 fixed initial states for each of the 40 tasks. Our RoboMME harness reproduces the reported MME-VLA result on $\pi_{0.5}$ within 0.1 percentage points on one evaluation seed. On RMBench, training uses the full benchmark but evaluation only the first ten of the 100 official episodes per task, a reduced protocol under which the authors' released memory-free $\pi_{0.5}$ checkpoint reproduces its published result (Chen et al., 2026b). Ablations use a reduced RoboMME protocol: two tasks per family, eight in all, fixed before the campaign and evaluated on the first ten episodes of each task; the ablation models and their training are specified in Appendix B.1. *Real-world evaluation:* Policies are evaluated under a schedule of initial scenes shared by all methods, and a trial counts only if its memory-dependent part is correct: ten trials each on two of the ordering tasks (e.g. Figure 2), and on 25 pick up N times trials, five per value of N (Appendix A.4).

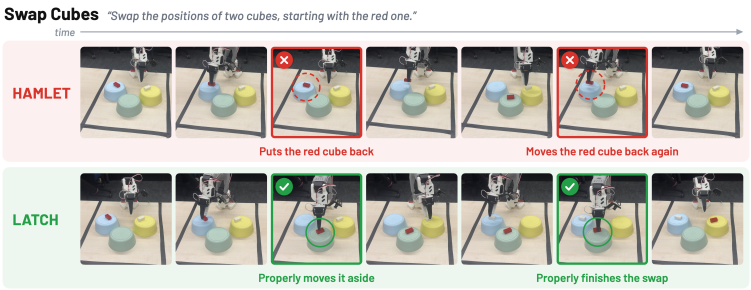


Figure 2: **Real-world rollout:** Executed by LATCH (ours) and HAMLET (Koo et al., 2026). LATCH succeeds, while HAMLET is confused about next steps after the first pick-up.

4.2 Results and Discussion

Results on simulated benchmarks. LATCH achieves the highest average success on RoboMME on both backbones (Table 1). On $\pi_{0.5}$ it reaches 47.2%, improving on the matched uniform-memory baseline MME-VLA (Dai et al., 2026) by 2.7 points and on the annotated hybrid MemER (Sridhar et al., 2026) by 4.8 without any annotation, with its largest gains in the Counting and Imitation families; MemER keeps the lead in Permanence and Reference, where a language subtask names the object to recall. On GR00T N1.6 the ordering carries over: LATCH improves on MME-VLA, HAMLET (Koo et al., 2026), and the memory-free backbone, with its margin in Counting. On RMBench (Table 2) LATCH is again the strongest perceptual memory, driven by the M(1) tasks, and it matches the annotated hybrid Mem-0 (Chen et al., 2026b) on the M(n) tasks and on the nine-task mean, trailing it only on M(1); the M(n) tasks remain hard for every method. On LIBERO (Table 3), LATCH keeps its backbone's performance to within about the seed variance, so the memory adds no cost on regular tasks.

Results on the real robot. On the SO-101 arm, LATCH solves the two ordering tasks in 8 and 10 of ten trials and keeps an exact count on "pick up" more often than any baseline, in 13 of 25 trials against 8 for MME-VLA, 3 for HAMLET, and 2 for the memory-free $\pi_{0.5}$. Results are reported in

Table 1: **RoboMME results across two VLA backbones.** Success rate (%); indented rows share the backbone above. †: reported in RoboMME (Dai et al., 2026); others: mean of three evaluation seeds, std on Overall. Bold, underline: best and second best per column and backbone.

Method	Memory	Type	Annot.	Count.	Perm.	Ref.	Imit.	Overall ↑
Human†	—	—	—	88.5	91.0	93.0	89.5	90.5
MemoryVLA† (Shi et al., 2026)	cognitive bank	perceptual	×	7.2	8.3	6.2	6.8	7.1
SAM2Act+† (Fang et al., 2025)	spatial features	perceptual	×	35.3	26.0	16.8	7.3	21.4
$\pi_{0.5}$† (Physical Intelligence et al., 2025)	none	none	×	28.8	17.0	17.2	8.8	17.9
$\pi_{0.5}$ + action history†	action history	actions	×	29.1	22.8	15.9	11.2	19.7
HAMLET (Koo et al., 2026)	moment tokens	perceptual	×	32.8	24.5	22.3	19.8	24.9 ± 1.4
MemER† (Sridhar et al., 2026)	keyframes + subtasks	percept. + symbolic	✓	48.8	**53.2**	**38.0**	29.5	42.4
MME-VLA† (Dai et al., 2026)	uniform frames	perceptual	×	65.2	25.1	36.3	51.4	44.5
LATCH (ours)	gated uniform frames	perceptual	×	**69.8**	28.2	37.3	**53.7**	**47.2 ± 0.9**
GR00T N1.6 (NVIDIA et al., 2025)	none	none	×	2.5	0.5	6.5	4.3	3.5 ± 0.3
HAMLET	moment tokens	perceptual	×	21.3	17.8	**16.0**	9.5	16.2 ± 0.2
MME-VLA	uniform frames	perceptual	×	34.2	**19.0**	**16.0**	16.7	21.5 ± 0.1
LATCH (ours)	gated uniform frames	perceptual	×	**45.2**	16.8	14.8	**17.5**	**23.6 ± 0.3**

Type: representation supplied to the policy. *Annot.*: training requires manual subtask or keyframe annotations.

Table 2: **RMBench results with a shared $\pi_{0.5}$ backbone.** Success rate (%) on five M(1) and four M(n) dual-arm tasks; first 10 of 100 official episodes per task, mean ± std of three seeds. Mem-0, on the benchmark authors' own architecture, is re-run in our harness; its unreleased M(n) planner is re-trained with their recipe, so its M(n) score is a lower bound (Appendix A.3). Columns as in Table 1; bold: best per column.

Method	Memory	Type	Annot.	M(1)	M(n)	All
Mem-0 (Chen et al., 2026b)	planner + anchor/sliding	percept. + symbolic	✓	**55.3 ± 3.1**	15.8 ± 15.3	**37.8 ± 6.8**
$\pi_{0.5}$	none	none	×	14.7 ± 7.0	5.0 ± 5.0	10.4 ± 6.1
HAMLET (Koo et al., 2026)	moment tokens	perceptual	×	21.3 ± 4.2	**18.3 ± 1.4**	20.0 ± 1.9
MME-VLA (Dai et al., 2026)	uniform frames	perceptual	×	35.3 ± 2.3	13.3 ± 8.0	25.6 ± 4.8
LATCH (ours)	gated uniform frames	perceptual	×	50.7 ± 4.6	**18.3 ± 6.3**	36.3 ± 3.6

Table 4. The per-N rows show that LATCH is exact in nine of ten trials up to $N{=}2$ and, when it misses, mostly picks past N and never stops short, whereas $\pi_{0.5}$ and HAMLET mostly overshoot and MME-VLA more often fails outright or stops short. During real-world deployment, we observed that upon successful completion of the task, HAMLET and MME-VLA tend to repeat the task once again. Especially when the memory frames buffer grows. Despite that MME-VLA achieves a comparable performance on the ordering tasks, LATCH does not attempt to repeat the task, revealing the significant capability to maintain long-term memory, most noticeable in the "swap" task. More details about the real-world setup are provided in Appendix A.4

Gate ablation. We ablate LATCH on the eight-task RoboMME subset, first asking what the gates contribute and then which design choices the result depends on. *Do the gates select useful frames?* At inference, we keep only the third of the memory frames that the trained gates rank highest, or only the third they rank lowest (Table 5); both interventions drop the same number of frames, so their difference isolates the ranking. The top-ranked frames retain clearly more success than the bottom-ranked ones, and they do so in every task family, so the gates rank the history by its use to the policy. *What do the query-aware read gate and the query-blind write gate prioritize?* Switching a gate off at inference is an out-of-distribution intervention on a policy trained with it, so we instead train one model per gate from the same warm start (Table 5). Any gate structure strengthens the memory where it is tested most directly: every gated configuration improves Counting over the ungated memory (Appendix B.1), and without the warm start the two gates together add about three points overall. The two gates also specialize (per task in Appendix B.1): the write gate, which weighs each frame by its content alone, is stronger on Permanence and on tasks where one stored moment carries the state, such as StopCube and the Unmask tasks, whereas the read gate, which lets the current action decide where to look, is stronger on Imitation and on tasks where the relevant frame changes as the episode unfolds, such as BinFill. With both gates, the policy comes within two points of the better single gate in every family and exceeds both in Counting, so the two gates combine rather than compete; at ten episodes per task these differences are of the order of the seed spread, and we read them as tendencies of the gates rather than settled rankings.

Design-choice ablation. The warm start from the uniform-memory policy is the largest single factor: training the memory-conditioned policy from the base weights instead costs more than 20

Table 3: LIBERO mean success (%); per suite in Table 9.

Policy	Mean
$\pi_{0.5}$	97.1 ± 0.2
HAMLET	97.4 ± 0.2
MME-VLA	97.3 ± 0.6
LATCH (ours)	97.0 ± 0.5
GR00T N1.6	96.8 ± 0.4
HAMLET	96.7 ± 0.7
MME-VLA	96.1 ± 0.3
LATCH (ours)	95.6 ± 0.7

Table 4: Real-world SO-101 results (combined models): successes over trials on the memory-dependent part; pick-up rows show the five trials per N (✓ exact, ↑ past N, ↓ short, × failed).

Task	$\pi_{0.5}$	HAMLET	MME-VLA	LATCH (ours)
Return Lego to original side	1/10	1/10	8/10	8/10
Swap Lego starting with red	2/10	3/10	10/10	10/10
Pick up Lego N times, exact	2/25	3/25	8/25	**13/25**
N=1	↑×↑×✓	✓↑✓↑✓	✓××××✓	✓✓✓✓✓
N=2	××↓✓↑	↑↑×↑↑	××✓✓✓	✓✓✓✓×
N=3	××↑↓×	↑↑↑↑↓	↑×↑✓✓	✓↑✓↑↑
N=4	↑↑↑↑↓	↑××↓↑	××↑↑↑	↑×↑↑↑
N=5	↑↑↑↑↑	↑×↑↑↑	↓↓↓✓↓	✓↑✓↑↑

Table 5: **Ablations on the eight-task RoboMME subset.** Success rate (%), with the standard deviation over three seeds on the overall. Each block varies one design choice; the full model in the last row is the reference for every block except frame selection.

Variant	Count.	Perm.	Ref.	Imit.	Overall
Gates, trained from the warm start					
Write gate only	50.0	**41.7**	35.0	74.6	50.3 ±4.0
Read gate only	**51.7**	36.7	**36.7**	**80.0**	**51.2 ±1.2**
Frame selection at inference, trained LATCH					
Top-ranked third	**36.7**	**35.0**	**36.7**	**43.9**	**38.1 ±2.8**
Bottom-ranked third	28.3	31.7	33.3	35.4	32.2 ±0.8
Without the warm start (from base $\pi_{0.5}$)					
No gate	26.7	35.0	**28.3**	23.3	28.3 ±4.0
Write + read gates	**30.0**	**46.7**	21.7	**26.7**	**31.2 ±3.3**
LATCH (full model)	55.0	40.0	35.6	80.0	52.6 ±3.4

Variant	Count.	Perm.	Ref.	Imit.	Overall
Memory budget, write + read gates					
8 frames	41.7	35.6	30.0	56.7	41.0 ±4.0
16 frames	43.3	32.0	28.3	66.3	42.5 ±1.4
32 frames (full model)	**55.0**	**40.0**	**35.6**	**80.0**	**52.6 ±3.4**
64 frames	45.7	35.0	31.7	67.6	45.0 ±1.7
128 frames	45.0	33.3	16.7	56.7	37.9 ±0.7
LATCH (full model)	55.0	40.0	35.6	80.0	52.6 ±3.4

points, with or without the gates (Table 5). The memory budget follows an inverted U (Table 5): success rises from 8 to 32 frames and then falls at 64 and 128 (4 frames, in Appendix B.1, is lower still), and 32 frames is the best budget in every task family, so beyond 32 frames additional history dilutes the evidence the policy needs rather than adding to it. The memory costs about half a policy call and a fixed amount of GPU memory whatever its budget, and the gates add a few milliseconds per call over the uniform memory they build on (Appendix B.2).

How does LATCH compare with perceptual memory methods? Across the diverse memory-intensive tasks of RoboMME, LATCH is the strongest perceptual memory on both backbones, ahead of the matched uniform-memory baseline MME-VLA (Dai et al., 2026) and of HAMLET (Koo et al., 2026), and its gains over the uniform memory come from the Counting and Imitation tasks (Appendix B), where the evidence is the robot's own motion over the whole episode. HAMLET's compressed recent window, the own-architecture memories of MemoryVLA (Shi et al., 2026) and SAM2Act+ (Fang et al., 2025), and an action history alone all fall well behind, so what matters is retaining visual evidence over the whole episode in a form the action expert can attend to, and LATCH's action-gated readout is the best instance of it.

How does it compare with hybrid memory methods that require additional manual annotation? On RoboMME an annotation-free perceptual memory exceeds the annotated hybrid on average: LATCH is ahead of MemER (Sridhar et al., 2026), whose VLM predicts language subtasks from retrieved keyframes and is trained on subtask annotation, and the two divide the benchmark. LATCH leads where the evidence is the robot's own motion, its timing, or a count accumulated over the episode, which a uniformly sampled history covers; MemER leads where one earlier observation has to be recalled as a nameable fact, an object under a cup or a highlighted target, which a language subtask carries in a few tokens. On RMBench the same split appears against Mem-0 (Chen et al., 2026b), whose planner is trained on subtask annotation: it is ahead on the M(1) tasks by about one seed spread, level with LATCH on the M(n) tasks, where its seeds disagree widely, and within a seed spread of LATCH on all nine tasks, so the annotation-free memory matches the hybrid on this benchmark as a whole. The hybrids buy that advantage with annotation and a second model at inference; the perceptual memory pays only for its frames.

Is the memory used reliably, or can it confuse the policy? Added history can hurt an imitation policy: with naively stacked frames it copies its previous action or latches onto spurious temporal correlations (de Haan et al., 2019; Wen et al., 2020; Seo et al., 2023), long contexts degrade diffusion policies through the same shortcut (Torne et al., 2025), HAMLET reports that naive multi-

frame input lowers its backbone on LIBERO (Koo et al., 2026), and MemER finds language memories brittle on counting (Sridhar et al., 2026). Counting is the probe for this failure: the count exists only in memory, the frame after the third pick looks like the frame after the second, and every error is legible as a count that ran short, ran over, or was lost. LATCH passes it in every setting we tested: it leads the Counting family on RoboMME on both backbones, on the real arm it keeps an exact count more often than any baseline, including the uniform memory reading the same 32 frames, and its misses are late overshoots rather than lost counts; on LIBERO, where naive history is known to hurt, it costs nothing. The memory is therefore used as evidence of task progress rather than as a cue to repeat the last action.

Does it generalize across backbones? LATCH transfers to GR00T N1.6 with no change beyond the backbone's own recipe: the ordering of Table 1 is reproduced, LATCH ahead of MME-VLA, HAMLET, and the memory-free backbone, with its margin again in Counting. The absolute level is lower on this backbone and each arm is one training run, so the result establishes that the mechanism and its ordering carry over across architectures, with the size of the margin a property of the backbone.

Does it transfer across embodiments, datasets, and the real world? LATCH carries over to a dual-arm embodiment, a second memory benchmark, and a real arm. On RMBench it is the best perceptual memory and matches the annotated Mem-0 overall, with its gain in the M(1) tasks; the M(n) tasks, which need several observations accumulated over repeated attempts, remain open for every method at this protocol, and with ten episodes per task the study fixes the direction of the ranking rather than its size. On the real arm the two ordering tasks are solved in nearly every trial, so the perceptual memory carries the task state through a real episode, and on the counting task LATCH holds the count more often than MME-VLA.

Does it affect regular tasks? LIBERO is largely Markovian and near saturation, so it tests whether memory costs anything on tasks that do not need it. It does not: LATCH keeps its backbone's performance within about the seed variance on both backbones, where naive frame stacking is reported to lose several points (Koo et al., 2026), so the memory can stay switched on for regular tasks.

Which design choices matter, and at what cost? The result rests on a bounded perceptual memory at its 32-frame budget, warm-started from the uniform-memory policy, for about half a policy call. On this memory the gates add structure the policy uses: they rank the frames it needs, and the two gates specialize by the kind of evidence a task requires, for a few milliseconds per call.

Limitations and future work. LATCH stores raw frames sampled uniformly in time, and its limits follow from that choice. Where the task state is a single nameable fact, such as an object hidden under a cup, the annotated hybrids lead on the Permanence tasks, since a language subtask carries such a fact in a few tokens while a frame budget must keep the right frame and read it back. Perceptual and symbolic memory are therefore complementary, and the open problem is combining them without the annotation the hybrids rely on: deriving symbolic state from the same action signal that organizes the perceptual memory, and keeping memory at more than one time scale so that one decisive moment survives a long episode.

5 Conclusion

LATCH shows that a bounded perceptual memory inside a vision-language-action policy can be organized by the action objective alone. Starting from a uniformly sampled history, a read gate decides where the action queries look and a write gate how loudly each frame speaks, and both learn which past observations the current action needs without keyframe or subtask annotation or an external memory policy. The result is the strongest perceptual memory on the primary memory benchmark on both backbones and competitive with annotated hybrids; it carries over to a dual-arm embodiment and a real arm.

AI use statement

LLM-based assistants were used while preparing this paper. They helped edit the text for clarity and consistency, draft and revise parts of sections under the authors' direction, write scripts that produce figures and tables from the logged results, and format references. Coding agents were used

to implement and debug training, evaluation, and baseline code and to run experiments; the authors reviewed this code and checked the reported results against the experiment logs. The authors take full responsibility for the content of this paper.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-VL technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, 2023. URL https://arxiv.org/abs/2307.15818.

Qingwen Bu, Yan Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Learning to act anywhere with task-centric latent actions. In *Robotics: Science and Systems*, 2025. URL https://arxiv.org/abs/2505.06111.

Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, Steven Palma, Pepijn Kooijmans, Michel Aractingi, Mustafa Shukor, Dana Aubakirova, Martino Russi, Francesco Capuano, Caroline Pascal, Jade Choghari, Khalil Meftah, Maxime Ellerbach, Jess Moss, and Thomas Wolf. LeRobot: State-of-the-art machine learning for real-world robotics in PyTorch. https://github.com/huggingface/lerobot, 2024.

Jingjing Chen, Hongjie Fang, Chenxi Wang, Shiquan Wang, and Cewu Lu. History-aware visuomotor policy learning via point tracking. In *IEEE International Conference on Robotics and Automation*, 2026a. URL https://arxiv.org/abs/2509.17141.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation, 2025. URL https://arxiv.org/abs/2506.18088.

Tianxing Chen, Yuran Wang, Mingleyang Li, Yan Qin, Hao Shi, Zixuan Li, Yifan Hu, Yingsheng Zhang, Kaixuan Wang, Yue Chen, Hongcheng Wang, Junjie Wang, Tianhang Yang, Renjing Xu, Ruihai Wu, Yao Mu, Yaodong Yang, Hao Dong, and Ping Luo. RMBench: Memory-dependent robotic manipulation benchmark with insights into policy design, 2026b. URL https://arxiv.org/abs/2603.01229.

Egor Cherepanov, Nikita Kachaev, Alexey K. Kovalev, and Aleksandr I. Panov. Memory, benchmark & robots: A benchmark for solving complex tasks with reinforcement learning, 2026. URL https://arxiv.org/abs/2502.10550.

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, 2026. URL https://arxiv.org/abs/2603.04639. Version 3, 2026-05-26.

Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/947018640bf36a2bb609d3557a285329-Abstract.html.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. LongRoPE: Extending LLM context window beyond 2 million tokens. In *International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2402.13753.

Haoquan Fang, Markus Grotz, Wilbert Pumacay, Yi Ru Wang, Dieter Fox, Ranjay Krishna, and Jiafei Duan. SAM2Act: Integrating visual foundation model with a memory architecture for robotic manipulation. In *International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2501.18564.

Gemini Robotics Team, Abbas Abdolmaleki, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Ashwin Balakrishna, Nathan Batchelor, Alex Bewley, Jeff Bingham, et al. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer, 2025. URL https://arxiv.org/abs/2510.03342.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs), 2016. URL https://arxiv.org/abs/1606.08415.

Huiwon Jang, Sihyun Yu, Heeseung Kwon, Hojin Jeon, Younggyo Seo, and Jinwoo Shin. ContextVLA: Vision-language-action model with amortized multi-frame context, 2025. URL https://arxiv.org/abs/2510.04246.

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. In *Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, 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*, 2026. URL https://arxiv.org/abs/2510.00695.

Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Raymond Yu, Caelan Reed Garrett, Fabio Ramos, Dieter Fox, Anqi Li, Abhishek Gupta, and Ankit Goyal. HAMSTER: Hierarchical action models for open-world robot manipulation. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2502.05485. URL https://arxiv.org/abs/2502.05485.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2306.03310.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 2022. URL https://arxiv.org/abs/2112.03227.

Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. ManiSkill: Generalizable manipulation skill benchmark with large-scale demonstrations. In *Advances in Neural Information Processing Systems, Track on Datasets and Benchmarks*, 2021.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. RoboCasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems*, 2024. URL https://arxiv.org/abs/2406.02523.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Jim Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia

Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots, 2025. URL https://arxiv.org/abs/2503.14734.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI Conference on Artificial Intelligence*, 2018.

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. In *Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40, 2025. doi: 10.48550/arXiv.2504.16054. URL https://arxiv.org/abs/2504.16054.

Seokin Seo, HyeongJoo Hwang, Hongseok Yang, and Kee-Eung Kim. Regularized behavior cloning for blocking the leakage of past action information. In *Advances in Neural Information Processing Systems*, volume 36, pp. 2128–2153, 2023.

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. doi: 10.48550/arXiv.2508.19236. URL https://arxiv.org/abs/2508.19236.

Lucy Xiaoyang Shi, Brian Ichter, Michael 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 *International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 54919–54933, 2025. doi: 10.48550/arXiv.2502.19417. URL https://arxiv.org/abs/2502.19417.

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. doi: 10.48550/arXiv.2510.20328. URL https://arxiv.org/abs/2510.20328.

Marcel Torne, Andy Tang, Yuejiang Liu, and Chelsea Finn. Learning long-context diffusion policies via past-token prediction. In *Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, 2025. doi: 10.48550/arXiv.2505.09561. URL https://arxiv.org/abs/2505.09561.

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, 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*, volume 33, pp. 2564–2575, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1b113258af3968aaf3969ca67e744ff8-Abstract.html.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11097–11107, 2020.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *IEEE/CVF International Conference on Computer Vision*, pp. 11975–11986, 2023.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

A Experiment Setup

A.1 Implementation Details

Memory and gates. LATCH is instantiated on $\pi_{0.5}$ (Physical Intelligence et al., 2025) and GR00T N1.6 (NVIDIA et al., 2025). The memory samples $N{=}32$ frames uniformly in time from the observation history and encodes each with the frozen vision encoder into $P{=}16$ tokens, a budget of $B{=}512$ tokens. The write gate is a two-layer perceptron of hidden width 512 on the frame summary, and the read gate one of hidden width 256 on the action and frame summaries; together they add about 1.3M parameters, three orders of magnitude fewer than the policy. Both gates are zero-initialized ($b{=}0$, $g{=}1$), so the gated policy equals the uniform-memory policy at the start of the gate phase.

Optimization. Unless a benchmark paragraph below states otherwise, all policies use AdamW (Loshchilov & Hutter, 2019) ($\beta_1{=}0.9$, $\beta_2{=}0.95$), gradient clipping at 1.0, an exponential moving average of the parameters at rate 0.999, and a frozen vision tower; the gate phase uses a cosine schedule with 5% warm-up from a peak learning rate of $2{\times}10^{-5}$ to $2{\times}10^{-6}$ and weight decay 10^{-10}, and the action loss is the only training signal. Per-benchmark steps, batch sizes, initializations, and evaluation protocols are given in the following subsections.

A.2 Baselines

The comparison separates methods by the memory available to the policy: the memory-free backbones bound what the current observation achieves; MME-VLA is the nearest perceptual memory and, since LATCH's first phase trains the same uniform-memory policy with the same recipe, the control that isolates the gates; HAMLET is the strongest published perceptual memory with released code for both backbones and represents learned compression of the recent history rather than a fixed buffer; the remaining RoboMME rows are taken from the benchmark's own comparison and cover representations we do not retrain; and Mem-0 is the RMBench authors' policy. Baselines train for the total step budget of Table 6 under their published recipes: on $\pi_{0.5}$, MME-VLA is the checkpoint released by the RoboMME authors (Dai et al., 2026), and HAMLET is trained with the same recipe and its released configuration (four moment tokens over a history of 16 steps, two views, full finetuning); on GR00T N1.6, MME-VLA is trained with the HAMLET recipe. On GR00T N1.6, the HAMLET row uses a publicly uploaded RoboMME checkpoint trained with the authors' released recipe (60,000 steps, batch size 32, peak learning rate 10^{-4}). The memory-free GR00T row is a public GR00T N1.6 checkpoint trained on RoboMME with the same recipe and the memory module off. Reported rows are the mean of three checkpoints and three seeds in RoboMME.

A.3 Training and Evaluation Protocols

Table 6 lists, per benchmark, the demonstrations, the initialization and step budget of each phase, the batch size, and the evaluation protocol; every row uses evaluation seeds 0, 7, and 42.

Table 6: Training and evaluation per benchmark. Control steps are the budget of the uniform-memory policy (MME-VLA) and of the other baselines; LATCH steps are its uniform-memory phase plus its gate phase. Every evaluation uses three seeds.

Benchmark	Demonstrations	Initialization	Control steps	LATCH steps (uniform + gate)	Batch	Evaluation
RoboMME, $\pi_{0.5}$	100 per task (1,600)	$\pi_{0.5}$ base	80k	50k + 30k	128	50 episodes per task (800)
RoboMME, GR00T N1.6	100 per task (1,600)	GR00T N1.6 base	60k	40k + 20k	32	50 episodes per task (800)
LIBERO, $\pi_{0.5}$ and GR00T N1.6	50 per task (2,000)	$\pi_{0.5}$ / GR00T N1.6 base	30k	21k + 9k	64	50 initial states per task (2,000)
RMBench, $\pi_{0.5}$	50 per task (450)	$\pi_{0.5}$ base	30k	21k + 9k	64	first 10 of 100 episodes per task (90)
Ablations (RoboMME subset)	1,600 (data-fraction arms: 800 and 400)	released MME-VLA checkpoint (80k)	80k + 15k	80k + 15k (Appendix B.1)	128	first 10 episodes of 8 tasks (80)
Real robot, $\pi_{0.5}$	50, 50, 100; combined 200	$\pi_{0.5}$ base	3k–8k to plateau	same, no uniform phase	64	10, 10, and 25 trials

RoboMME. On $\pi_{0.5}$, MME-VLA is the RoboMME authors' released checkpoint, trained with their recipe for 80,000 steps at batch size 64 with a learning rate of $5{\times}10^{-5}$ held constant after 10,000 warm-up steps, an action horizon of 20 in joint space, no proprioceptive input, and the SigLIP (Zhai et al., 2023) vision tokens frozen and cached; HAMLET is trained with the same recipe. LATCH's first phase is our own run of the same recipe for 50,000 steps, and its gate phase then runs 30,000 steps at batch size 128 with a cosine schedule from $2{\times}10^{-5}$ to $2{\times}10^{-6}$ after 1,500 warm-up steps, on four A100 GPUs in about 14 hours. The GR00T N1.6 policies follow

the HAMLET recipe on four H100 GPUs: 60,000 steps at a global batch size of 32 with AdamW (weight decay 10^{-5}, no moving average), a peak learning rate of 10^{-4} with cosine decay after 5% warm-up, an action horizon of 16, two views at 224×224, and the top four language-model layers, the projector, the diffusion head, and the vision-language layer norms trainable with the rest frozen; LATCH branches from the uniform-memory policy at 40,000 steps for 20,000 gate steps at a peak of 4×10^{-5}. Evaluation runs all 50 episodes of each of the 16 tasks with the benchmark's held-out initializations and success definitions, executing 16 actions per replan under a cap of 1,300 environment steps after which an episode counts as a failure; rollouts run on eight A100 GPUs with 32 parallel workers and evaluation seeds 0, 7, and 42, and the first ten episodes per task serve model selection.

LIBERO. All rows train for 30,000 steps at an effective batch size of 64 on four GH200 GPUs: on $\pi_{0.5}$ with a cosine schedule from 5×10^{-5} to 5×10^{-6} after 1,500 warm-up steps, and on GR00T N1.6 with the RoboMME GR00T recipe above at a peak of 10^{-4}; LATCH branches from the uniform-memory checkpoint at 21,000 steps for 9,000 gate steps at the gate-phase schedule of Appendix A.1 on $\pi_{0.5}$ and a peak of 4×10^{-5} on GR00T N1.6. Evaluation, on four GH200 GPUs with 12 workers, uses 50 fixed initial states for each of the 40 tasks, 2,000 episodes per seed, an action chunk of 20 replanned every 16 steps, and episode caps of 220, 280, 300, and 520 steps for the Spatial, Object, Goal, and Long suites; it reports suite-average success and its mean over the four suites.

RMBench. RMBench runs in RoboTwin 2.0 (Chen et al., 2025) on the SAPIEN engine (Xiang et al., 2020) with the Aloha-AgileX embodiment, two 6-DoF arms. Its nine tasks are graded by task memory complexity: the five M(1) tasks (Observe and Pick Up, Rearrange Blocks, Put Back Block, Swap Blocks, Swap T) require one past observation or a fixed short window, and the four M(n) tasks (Blocks Ranking Try, Press Button, Cover Blocks, Battery Try) require several observations accumulated over repeated attempts or trial and error. Mem-0 is the benchmark authors' policy: a Qwen3-VL (Bai et al., 2025) planning module that predicts the next subtask from the initial observation, the goal, and a key memory of completed subtasks, a subtask-end classifier, and a diffusion execution policy with anchor and sliding memories; it is trained with the subtask annotations that RMBench provides for every action–observation pair. All rows train on four GH200 GPUs at batch size 64 with an exponential moving average of 0.999 and a cosine schedule with 5% warm-up decaying to a tenth of the peak: MME-VLA and HAMLET for 30,000 steps from the base $\pi_{0.5}$ weights at a peak of 5×10^{-5}, and LATCH from the MME-VLA checkpoint at 21,000 steps for 9,000 gate steps at a peak of 2×10^{-5}; HAMLET uses four moment tokens over a history of 16 steps, three camera views, and full finetuning. Evaluation, on A100 nodes, uses the first ten of the 100 official episodes per task with unseen instructions, an action chunk of 20 executed for 16 steps, and the benchmark's per-task step limits (250 to 3,500) in a SAPIEN 3.0.0b1 harness, in which the authors' released memory-free $\pi_{0.5}$ checkpoint scores 10.0% on seed 7 against the published 10.4%. Mem-0 is re-evaluated in the same harness and protocol: its M(1) policies use the authors' released weights and reproduce their result (55.3% against 56.6% reported on 100 episodes), whereas its M(n) planner was never released, so we re-trained it with the authors' recipe, and its M(n) score (15.8% against the reported 28.5%) is a lower bound.

A.4 Real-World Setup

Robot and observations. All tasks use an SO-101 arm (Cadene et al., 2024) with a fixed front camera and a wrist camera (640×480 each), a six-dimensional joint state in absolute degrees, and 30 Hz control under a torque cap. The policy receives the language instruction, the current front and wrist images, the robot state, and the sampled perceptual history; it predicts a chunk of 20 actions, executes 16, and replans. Below are the parameters of the real-world setup, with visualizations in Figure 3

- Front Camera: Logitech C920 PRO HD Webcam.
- Wrist Camera: Innomaker U20CAM-720P.
- Workspace Dimensions: 36x47 cm, the camera center is at 13.5 cm from the upper-right workspace corner, at the height of 43.5 cm.
- Object set: red and white LEGO bricks, three colored plates for ordering tasks.

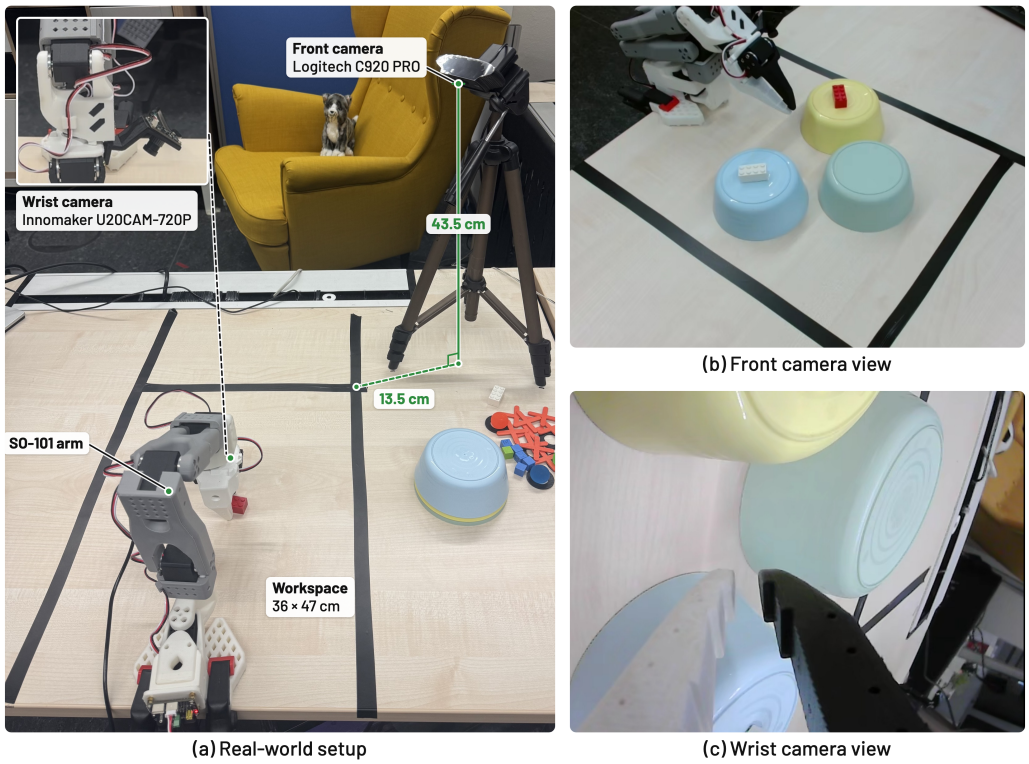


Figure 3: Real-world workspace setup.

Task definitions. Every task requires state that the current frame does not reveal. *Pick up the Lego and place it on the opposite side, and then return it to the original side*: pick up a Lego block, place it on the opposite side, and return it to the side it started on; success requires the final placement on the original side. *Swap the positions of two cubes, starting with the red one*: swap the positions of two Lego blocks, starting with the red one; success requires the final configuration and the correct first pick. *Pick up the Lego N times*: pick up and release the Lego block exactly $N \in \{1, \ldots, 5\}$ times, with N given in the instruction; success requires the exact count. A *combined* model is trained on all seven instructions. Instructions are lowercase with a trailing period, exactly as recorded. The visualization of the tasks is presented in Figure 4.

Training. The three task sets contain 50, 50, and 100 demonstrations (20 per value of N), and the combined set all 200 (87,045 frames, of which 5.6% belong to the observation phase and receive no action loss). For each set, a memory-free $\pi_{0.5}$, MME-VLA, HAMLET, and LATCH are trained from the stock $\pi_{0.5}$ base weights with one training seed on four GH200 GPUs at an effective batch size of 64, with a cosine schedule from 5×10^{-5} to 5×10^{-6} after 100 warm-up steps, an exponential moving average of 0.99, and no frozen parameters, so the vision encoder is finetuned as well; HAMLET uses four moment tokens over a history of 16 steps and both camera views. Unlike the simulation policies, the real-robot LATCH has no uniform-memory warm start: its memory and gates are trained jointly with the policy from the base weights, so that all four arms share one initialization. Every arm trains on all demonstrations to a training-loss plateau (less than 10% improvement over the last 500 steps, or a loss below 0.0025), which took 3,000 steps on the two 50-demonstration sets, 4,000 to 6,000 on Pick up Lego N times, and 8,000 steps (about six epochs) on the combined set; only the final checkpoint is kept. There is no held-out demonstration set, so training losses (0.0016–0.0021 on the combined set) rank nothing; rollouts decide.

Evaluation protocol. The combined model of each method is evaluated for ten trials on each ordering task and 25 on Pick up Lego N times, five per value of N, under a schedule of initial scenes, object placements, and values of N shared by all methods; a trial counts as a success only if

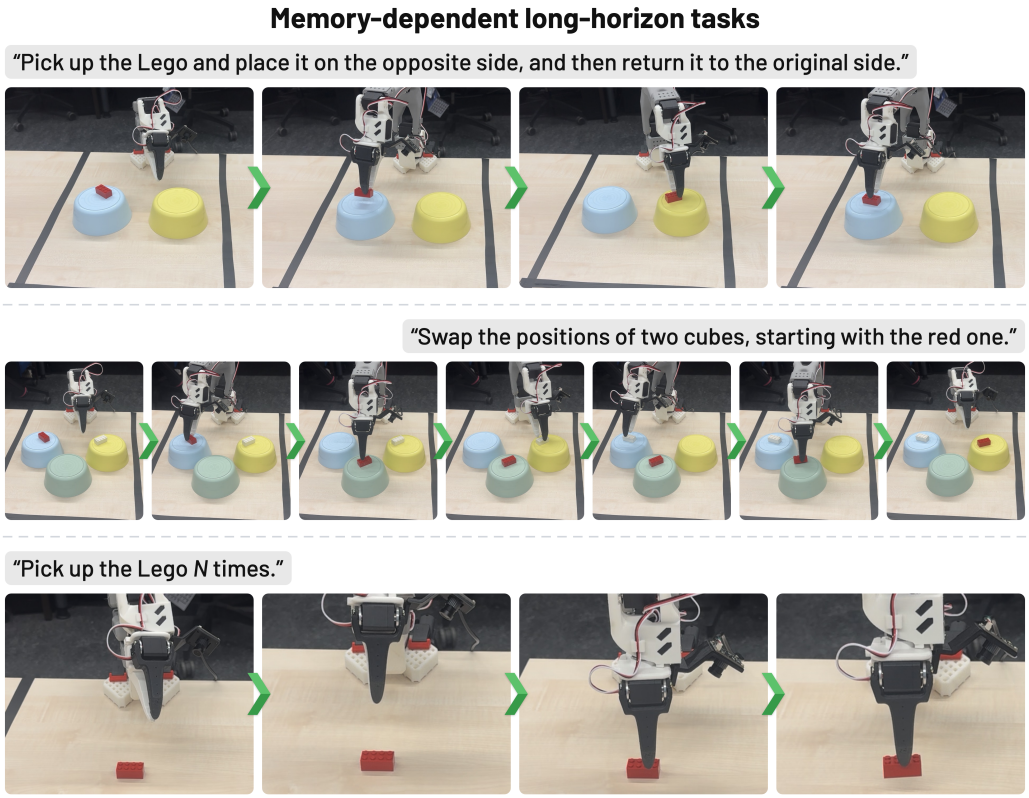


Figure 4: LATCH successfully executing three real-world tasks on SO-101 arm.

its memory-dependent part is correct. Results are reported in Table 4 of the main text; the single-task models were trained but not rolled out.

B Additional Results

Table 7: RoboMME success (%) on every task for the memory-free backbone (Base), HAMLET, MME-VLA, and LATCH on $\pi_{0.5}$ and GR00T N1.6: mean ± std over three evaluation seeds, 50 episodes per task and seed. † marks values reported in RoboMME (Dai et al., 2026), which gives a spread only for the overall mean (three checkpoints × three seeds). Italic rows are family means.

	$\pi_{0.5}$				GR00T N1.6			
Task	Base†	HAMLET	MME-VLA†	LATCH	Base	HAMLET	MME-VLA	LATCH
Counting	*28.8*	*32.8 ± 3.4*	*65.2*	*69.8 ± 0.3*	*2.5 ± 0.5*	*21.3 ± 1.0*	*34.2 ± 0.6*	*45.2 ± 2.9*
BinFill	30.0	30.7 ± 4.2	39.6	38.7 ± 2.3	2.0 ± 0.0	22.0 ± 5.3	18.0 ± 2.0	14.0 ± 2.0
PickXtimes	42.9	46.7 ± 2.3	87.3	94.7 ± 3.1	1.3 ± 1.2	26.7 ± 1.2	23.3 ± 4.2	55.3 ± 4.2
SwingXtimes	35.6	37.3 ± 1.2	92.0	97.3 ± 1.2	0.0 ± 0.0	30.0 ± 2.0	50.0 ± 2.0	62.7 ± 1.2
StopCube	6.7	16.7 ± 12.7	42.0	48.7 ± 2.3	6.7 ± 1.2	6.7 ± 4.6	45.3 ± 2.3	48.7 ± 6.4
Permanence	*17.0*	*24.5 ± 1.0*	*25.1*	*28.2 ± 0.3*	*0.5 ± 0.5*	*17.8 ± 1.3*	*19.0 ± 1.5*	*16.8 ± 2.3*
VideoUnmask	20.4	30.7 ± 4.2	32.7	39.3 ± 3.1	0.7 ± 1.2	15.3 ± 5.0	16.7 ± 1.2	18.7 ± 4.2
ButtonUnmask	22.2	24.7 ± 1.2	25.1	22.0 ± 2.0	0.0 ± 0.0	18.7 ± 3.1	20.0 ± 4.0	11.3 ± 5.0
VideoUnmaskSwap	18.7	24.0 ± 2.0	24.4	29.3 ± 3.1	1.3 ± 1.2	18.0 ± 2.0	23.3 ± 4.6	20.7 ± 3.1
ButtonUnmaskSwap	6.7	18.7 ± 1.2	18.2	22.0 ± 6.9	0.0 ± 0.0	19.3 ± 2.3	16.0 ± 9.2	16.7 ± 2.3
Reference	*17.2*	*22.3 ± 1.3*	*36.3*	*37.3 ± 1.5*	*6.5 ± 1.3*	*16.0 ± 1.8*	*16.0 ± 0.5*	*14.8 ± 2.1*
PickHighlight	11.3	23.3 ± 1.2	22.9	30.0 ± 2.0	0.0 ± 0.0	12.0 ± 4.0	10.0 ± 2.0	6.0 ± 0.0
VideoRepick	0.4	12.0 ± 4.0	30.4	26.7 ± 3.1	0.0 ± 0.0	13.3 ± 3.1	7.3 ± 3.1	7.3 ± 1.2
VideoPlaceButton	31.1	30.7 ± 2.3	60.0	54.0 ± 0.0	17.3 ± 2.3	22.7 ± 3.1	28.7 ± 4.2	28.7 ± 6.4
VideoPlaceOrder	25.8	23.3 ± 2.3	32.0	38.7 ± 3.1	8.7 ± 4.2	16.0 ± 3.5	18.0 ± 3.5	17.3 ± 4.2
Imitation	*8.8*	*19.8 ± 1.0*	*51.4*	*53.7 ± 1.8*	*4.3 ± 1.0*	*9.5 ± 0.9*	*16.7 ± 1.0*	*17.5 ± 0.9*
MoveCube	26.0	58.0 ± 4.0	77.8	81.3 ± 1.2	5.3 ± 2.3	24.7 ± 1.2	28.0 ± 4.0	28.0 ± 2.0
InsertPeg	1.6	2.0 ± 0.0	7.6	7.3 ± 1.2	0.7 ± 1.2	2.0 ± 2.0	3.3 ± 1.2	6.0 ± 0.0
PatternLock	2.9	10.0 ± 2.0	53.6	58.0 ± 3.5	4.7 ± 1.2	5.3 ± 1.2	6.7 ± 2.3	3.3 ± 3.1
RouteStick	4.7	9.3 ± 3.1	66.7	68.0 ± 2.0	6.7 ± 3.1	6.0 ± 2.0	28.7 ± 3.1	32.7 ± 3.1
Overall	17.9 ± 0.6	24.9 ± 1.4	44.5 ± 0.8	47.2 ± 0.9	3.5 ± 0.3	16.2 ± 0.2	21.5 ± 0.1	23.6 ± 0.3

Table 8: RMBench success (%) on every task for the benchmark authors' Mem-0 (Chen et al., 2026b) on its own architecture, re-run in our harness with a re-trained M(n) planner, and for the memory-free backbone (Base, the authors' released checkpoint), HAMLET, MME-VLA, and LATCH on $\pi_{0.5}$: mean ± std over three evaluation seeds, ten episodes per task and seed. Italic rows are the M(1) and M(n) means.

		$\pi_{0.5}$			
Task	Mem-0	Base	HAMLET	MME-VLA	LATCH
M(1)	*55.3 ± 3.1*	*14.7 ± 7.0*	*21.3 ± 4.2*	*35.3 ± 2.3*	*50.7 ± 4.6*
Observe and Pick Up	3.3 ± 5.8	6.7 ± 11.5	6.7 ± 5.8	3.3 ± 5.8	13.3 ± 5.8
Rearrange Blocks	80.0 ± 0.0	6.7 ± 5.8	56.7 ± 15.3	20.0 ± 10.0	40.0 ± 17.3
Put Back Block	100.0 ± 0.0	26.7 ± 15.3	33.3 ± 11.5	50.0 ± 26.5	100.0 ± 0.0
Swap Blocks	76.7 ± 5.8	13.3 ± 15.3	3.3 ± 5.8	90.0 ± 0.0	90.0 ± 10.0
Swap T	16.7 ± 15.3	20.0 ± 0.0	6.7 ± 5.8	13.3 ± 15.3	10.0 ± 10.0
M(n)	*15.8 ± 15.3*	*5.0 ± 5.0*	*18.3 ± 1.4*	*13.3 ± 8.0*	*18.3 ± 6.3*
Blocks Ranking Try	6.7 ± 11.5	0.0 ± 0.0	53.3 ± 11.5	16.7 ± 5.8	43.3 ± 5.8
Press Button	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Cover Blocks	33.3 ± 35.1	0.0 ± 0.0	0.0 ± 0.0	30.0 ± 26.5	23.3 ± 15.3
Battery Try	23.3 ± 15.3	20.0 ± 20.0	20.0 ± 17.3	6.7 ± 5.8	6.7 ± 11.5
All nine tasks	37.8 ± 6.8	10.4 ± 6.1	20.0 ± 1.9	25.6 ± 4.8	36.3 ± 3.6

Table 9: Success on LIBERO per suite (%, 50 initial states per task, mean ± std over three seeds), with the memory methods listed under their backbone.

Policy	Spatial	Object	Goal	Long	Mean
$\pi_{0.5}$	96.9	98.6	97.7	95.3	97.1 ± 0.2
HAMLET	97.9	99.1	97.6	95.2	97.4 ± 0.2
MME-VLA	97.5	98.3	97.5	95.7	97.3 ± 0.6
LATCH	98.1	98.5	96.9	94.4	97.0 ± 0.5
GR00T N1.6	98.1	99.7	97.3	91.9	96.8 ± 0.4
HAMLET	96.7	99.4	97.2	93.4	96.7 ± 0.7
MME-VLA	96.8	98.5	96.3	92.9	96.1 ± 0.3
LATCH	94.7	99.4	96.2	92.2	95.6 ± 0.7

B.1 Ablations

Protocol. Every trained arm starts from the RoboMME authors' released 80,000-step MME-VLA checkpoint and trains on all 16 RoboMME tasks for 15,000 further steps on one GH200 node: batch size 128 with FSDP over four devices, a cosine schedule with 750 warm-up steps from 2×10^{-5} to 2×10^{-6}, the vision tower frozen, training seed 7, and the final checkpoint at step 14,999 for every result. The from-scratch arms start from the base $\pi_{0.5}$ weights; the budget is set at 16 tokens per frame, so 64 to 2,048 tokens hold 4 to 128 frames. The no-memory arm continues the memory-free $\pi_{0.5}$ baseline for the same 15,000 steps. The gate arms train the write gate alone, the read gate alone, or both gates, which is the full model, and three conditioning variants: a read gate that sees only the memory, a write gate that also sees the action summary, and a write gate per token rather than per frame. The budget arms train without and with both gates at 4 to 128 frames, and the data arms keep every second or fourth demonstration of each task. The frame-selection arms change the trained LATCH policy of Table 1 without retraining it: frame selection keeps the third of the 32 memory frames (11 frames) that the write gate scores highest or lowest, with a value scale of one on the kept frames and zero on the others. The intervention arms switch a gate off (b_i=0 or g_i=1), invert the read gate, or shuffle the gate values across frames, also at inference. Arms are evaluated on the first ten episodes of each task of an eight-task subset, two per family: BinFill and StopCube (Counting), ButtonUnmask and VideoUnmask (Permanence), VideoPlaceButton and PickHighlight (Reference), MoveCube and PatternLock (Imitation), fixed before the campaign. Subset values are not compared with 16-task scores. Each cell is 80 episodes and each arm 240 over evaluation seeds 0, 7, and 42; the 33 arms total 7,920 episodes; episodes that end in a simulator error are excluded from the denominator, and the eight tasks are averaged unweighted. Table 5 reports the arms by family.

B.2 Computational Cost

Table 10: Speed and memory of the perceptual memory on $\pi_{0.5}$. Left: LATCH by memory budget (one A100, median of 30 policy calls). Right: LIBERO policies at batch size 1 on one GH200; a replan is one history push plus one call, every 16 steps; training throughput on four GH200. GPUs were shared, so read ratios rather than absolute times.

LATCH memory budget		
Frames	Call (ms)	GPU (GB)
0 (no memory)	42.8	6.6
8	63.1	8.4
16	63.7	8.4
32	65.1	8.4
64	66.8	8.2

LIBERO policies			
Policy	Call (ms)	Replan (ms)	Train (steps/s)
$\pi_{0.5}$	46.1	46.1	2.20
+ MME-VLA	59.5	79.2	2.16
+ LATCH	67.4	87.8	2.07
+ HAMLET	78.2	83.6	0.51

The cost lies in having a memory, not in its size: the 32-frame memory adds about 22 ms per call and 1.8 GB over the memory-free policy, while growing the budget from 8 to 64 frames adds under 4 ms and no memory. The gates add about 8 ms per call and cost 4% of training throughput over the uniform memory; HAMLET has the slowest call but a cheaper replan, and trains about four times slower.