SELECTIVE RECALL FOR LATE-BOUND ROBOTIC MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

People encounter a stream of events throughout the day, yet a later request—such as fetching an object put away hours earlier—may depend on only a few relevant memories. Inspired by this everyday need for selective recall, we study how robots can use accumulated experience when the task is not known in advance. Existing manipulation-memory evaluations often reveal goals during history encoding, place key evidence at fixed temporal positions, or associate each history with a single task, leaving task-conditioned access to rich, task-agnostic experience insufficiently evaluated. We introduce **late-bound manipulation**: a robot first experiences a long, continuous history containing numerous candidate events interleaved with background interactions, and only a subsequent instruction determines which evidence is needed for action. We develop MOSAIC, which pairs identical histories and execution-start states with different instructions requiring evidence from different temporal intervals. Its shared-history construction enables controlled comparisons of behavior under different subsequent instructions. We further propose ReWAM, a selective-memory world-action model that combines compact global memory, instruction-informed retrieval of detailed historical chunks, and recent local context for video and action prediction. On seven RM-Bench tasks, ReWAM achieves 50.57% mean success, compared with 55.00% for dense-history conditioning and 44.71% for dense visual history with local action context. On the evaluated three- and six-target MOSAIC configurations, it achieves 93% and 45% success, compared with 36% and 26% for dense-history conditioning. In separate timing experiments at 256 historical video chunks, ReWAM achieves a $3.06\times$ full-cycle inference speedup over dense-history conditioning, with latency increasing by only 4.27% from 1 to 256 chunks. Together, MOSAIC and ReWAM provide a framework for studying task-conditioned recall and its performance–cost trade-offs in long-history manipulation.

1 INTRODUCTION

People may draw on hours of accumulated experience to complete a task, even when the information they need comes from only a few moments. This suggests a useful perspective on memory-dependent behavior: task-relevant evidence can be sparse within a long stream of experience. Robots operating continuously in the real world face a similar challenge. They accumulate observations and interactions over time, while the instruction that determines which past information matters may arrive only later.

Such delayed requests create a fundamental challenge for memory design. Before receiving the specific instruction, a robot cannot tailor its memory solely to that task: information discarded as irrelevant may become necessary for a future request. Although task-independent compression remains possible, the robot must preserve broad coverage of potentially useful experience and subsequently access the information relevant to the current instruction. The same history may therefore require different evidence to be recalled under different requests.

Existing manipulation-memory benchmarks evaluate important history-dependent capabilities (Chen et al., 2026; Dai et al., 2026), but do not fully isolate this setting. Revealing the goal during history encoding allows task-specific retention, placing evidence at fixed temporal positions permits fixed access strategies, and pairing each history with only one request limits evaluation of

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.

instruction-dependent recall. We study **late-bound manipulation**, where a robot accumulates history before receiving the specific instruction that determines which historical evidence is needed for action.

To investigate this capability, we introduce MOSAIC. It constructs long interaction histories containing multiple candidate events and intervening irrelevant activity, then pairs identical histories and execution-start states with different subsequent instructions. These instructions require information from different temporal intervals, so the policy must condition its behavior on the subsequent request.

Supporting long histories also requires efficient access: attending to every historical token increases prediction cost even when task-relevant evidence is sparse. We therefore propose ReWAM, a selective-memory world-action model whose instruction-informed intermediate queries select historical detail, complementing compact global memory and recent context for video and action prediction.

Our contributions are threefold. First, we formulate **late-bound manipulation**, a setting motivated by real-world operation in which the specific task instruction arrives after multiple experiences have been accumulated. Second, we introduce MOSAIC, a benchmark for instruction-dependent recall from long histories with multiple candidate events and irrelevant information. Third, we propose ReWAM, a selective-memory world-action model, and evaluate its manipulation performance and computational cost against the Dense and Local-A baselines. Direct latency measurements show that selective historical access substantially limits inference-time growth over the tested history lengths.

2 RELATED WORK

Benchmarks for Memory-Dependent Robotic Manipulation. Memory-dependent manipulation benchmarks isolate tasks for which the current observation is insufficient. MemoryBench (Fang et al., 2025) studies spatial recall, while MIKASA-Robo (Cherepanov et al., 2026) and RoboMME (Dai et al., 2026) cover broader memory demands and LIBERO-Mem (Chung et al., 2025) emphasizes object-history reasoning. RMBench (Chen et al., 2026) characterizes task memory complexity, and RoboMemArena (Lei et al., 2026) provides subtask and keyframe annotations. RoboMME-Interference (Rathi, 2026) further examines robustness to unrelated intervening experience. Our focus is on how the relevance of past experience changes with a subsequently revealed instruction. MOSAIC pairs shared histories and execution-start states with different requests, providing a controlled setting for history-dependent manipulation under subsequently revealed task goals.

Robot Policies with Memory Context. History-aware policies differ in what they retain and how retained information reaches control. BPP (Mark et al., 2026) selects keyframes through VLM-detected, task-defined events, whereas EventVLA (Yang et al., 2026a) learns when to commit visual evidence to a keyframe buffer. MemER (Sridhar et al., 2025) uses selected visual keyframes for hierarchical subtask generation. MemoryVLA (Shi et al., 2026) retrieves and consolidates perceptual and cognitive features, while MEM (Torne et al., 2026) combines short-term video representations with long-term language memory. Gated Memory Policy (Gao et al., 2026) learns when memory should contribute. RoboTTT (Jiang et al., 2026) encodes trajectory context through fast-weight updates, demonstrating test-time training as a robot-memory mechanism. LingBot-VA (Li et al., 2026) also conditions video–action prediction on cached visual history. Beyond policy-internal memory, STaR (Yuan et al., 2026a) constructs task-agnostic multimodal memory for later queries, and MessyMem (Banwasi et al., 2026) stores interaction outcomes and associated visual evidence for subsequent mobile-manipulation tasks. These approaches motivate efficient access to past experience. Our predictive manipulation policy combines compact global context, instruction-selected historical detail, and recent observations.

World Action Models. World Action Models (WAMs) augment action generation with future visual prediction for control and planning (Du et al., 2023; Kim et al., 2026). VideoVLA (Shen et al., 2025), LingBot-VA (Li et al., 2026), and DreamZero (Ye et al., 2026) adapt pretrained video generators for coupled visual and action prediction, with the latter two retaining historical context through autoregressive KV caching. Fast-WAM (Yuan et al., 2026b) retains video-prediction supervision during training while avoiding future-video generation at inference. Closely related memory-based

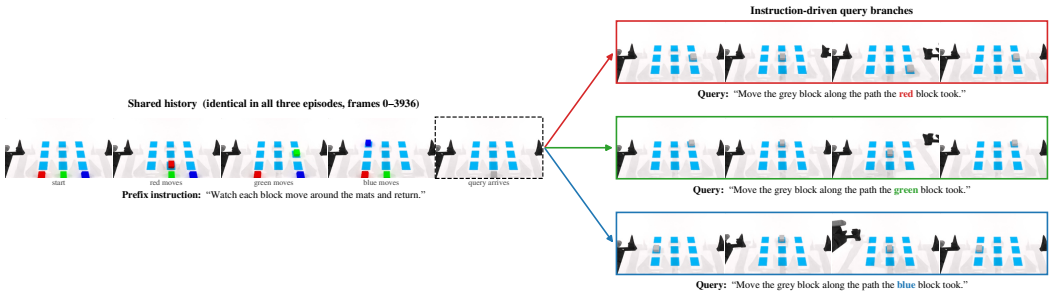


Figure 1: **Shared history with instruction-dependent action branches.** In Replay Block Trajectory, red, green, and blue blocks demonstrate routes at different times within a common history. The history-phase instruction does not specify which route will later be requested. At the query boundary, all branches share the same history and execution-start state. The subsequent instruction asks the robot to move the grey block along one of the demonstrated routes, making different historical intervals relevant to different requests. The branches illustrate the task protocol rather than measured model retrieval.

models include MemoryWAM (Yang et al., 2026b), which preserves initial and recent visual context while replacing older detailed tokens with learned gist tokens, and DiM-WAM (Wang et al., 2026), which maintains bounded banks of diverse event representations. Building on this direction, we investigate whether ReWAM can preserve manipulation performance over extended interaction histories while reducing the cost of accessing historical detail during video–action prediction.

3 Late-Bound Manipulation and the Benchmark

3.1 Problem Formulation

Let $H = (e_1, \ldots, e_T)$ denote a recorded interaction history, where each e_t contains the observations and any action or proprioceptive information exposed by the common policy interface. During history accumulation, the final instruction q is unavailable. A memory writer produces a memory state

$$M_T = W(H), \tag{1}$$

either incrementally or under an equivalent causal access constraint. At the query boundary, the robot receives q and the current observation o_0, and executes a policy conditioned on o_0, q, and M_T.

We construct a set of queries $\mathcal{Q}(H)$ for each history. Each query begins from the same simulator state s_0, but different queries can require different actions. For a query q, let E_q denote a directly sufficient set of historical evidence under the task rules. For many tasks, E_q occupies only a small part of H. These sets need not be unique: redundant observations or one-to-one constraints may allow alternative solutions.

Figure 1 illustrates this protocol: a subsequent instruction determines which previously demonstrated route the grey block should reproduce.

We evaluate policies by manipulation success and the computational cost of accessing retained history.

3.2 Construction Principles

Task-independent history formation. History generation and memory writing do not depend on the selected final query. A history-phase instruction may describe the ongoing activity, but does not identify the target of the subsequent request. For example, the instruction in Figure 1 asks the robot to observe all block trajectories without specifying which one to replay. The same history-phase instruction is used across branches, and the final query is withheld until the execution boundary.

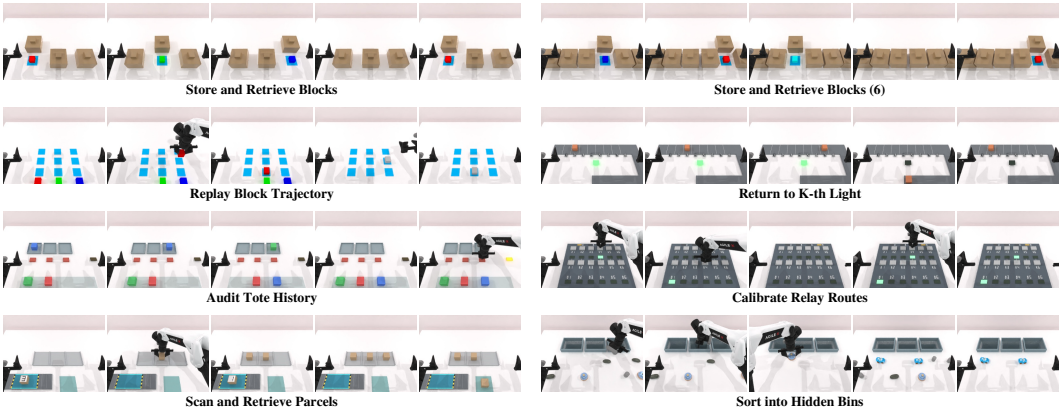


Figure 2: **Overview of MOSAIC.** Representative frames from eight task configurations spanning seven task families. Store and Retrieve Blocks includes three- and six-target variants; the remaining panels illustrate trajectory replay, ordinal event recall, historical conjunctions, relay composition, parcel identification, and demonstrated sorting. Each task accumulates history before the final instruction specifies the required action.

Shared-history instruction branches. Multiple final instructions are paired with a shared history and execution-start simulator snapshot (Figure 1). The current scene, robot configuration, and visible objects are identical across branches. Changing the instruction changes which historical evidence is needed and which action sequence should follow; query-specific initialization must not reveal the answer.

Transient or hidden evidence. Information needed for a task is available during the history but cannot be reliably recovered from the current observation alone. Examples include a disappeared package identifier, covered object contents, and a past button–light response.

Controlled interference. Candidate identities, event timing, and relevant mappings are randomized. We distinguish unrelated activity from similar competing evidence. Where possible, history length is increased while the direct target evidence and requested action remain unchanged.

3.3 TASK FAMILIES

MOSAIC comprises seven task families instantiated in eight configurations, including three- and six-target variants of Store and Retrieve Blocks. Figure 2 provides an overview of their scenes and representative interactions. The tasks span object–location associations, transient identity cues, demonstrated trajectories, composed mappings, conjunctions of historical facts, event ordering, and category-to-container rules.

All tasks follow the late-bound protocol: the final instruction is withheld during history accumulation, and different requests can make different parts of the same history relevant. Some requests rely on a localized observation, whereas others require combining multiple events or recovering their order. This variation tests selective recall beyond retrieving a single salient frame. Task descriptions and memory demands are provided in Appendix A.2.

We evaluate manipulation success on RMBench and MOSAIC and measure inference latency as retained history grows. Dataset and evaluation settings are reported in Appendix A.

4 REWAM: SELECTIVE-MEMORY WORLD-ACTION MODEL

The setting in Section 3 imposes two requirements on a policy. First, it must retain broad historical coverage because the final instruction is unavailable when experience is acquired. Second, it must use that instruction to access the evidence relevant to the current action. Keeping a detailed history provides broad coverage: for example, LingBot-VA maintains past video and action information

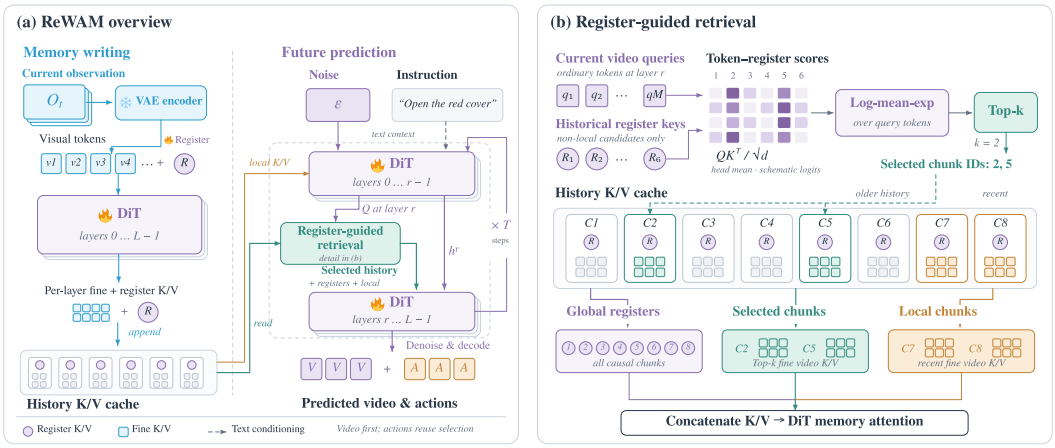


Figure 3: **ReWAM architecture and register-guided retrieval.** **(a)** Memory writing encodes observations with a register-augmented backbone and appends layer-specific detailed and register K/V states to the history cache. During prediction, instruction-informed video queries select historical chunks at layer r. Subsequent layers combine selected detail, global registers, and local context. Video prediction precedes action prediction, which reuses the video stage's final selection. **(b)** Query–register scores are averaged across heads and pooled across query tokens to select top-K chunks. Global register K/V, selected detailed video K/V, and recent detailed video K/V are combined for a single memory-attention operation. The illustrated selection uses $K = 2$ and excludes local chunks from ranking.

through a persistent key–value (KV) cache (Li et al., 2026). However, repeatedly attending to all detailed historical tokens increases the work performed at each prediction step as history grows, even when only a few events support the request. Section 5.4 evaluates the resulting latency under controlled history lengths.

These requirements motivate ReWAM, a selective-memory world-action model built on **LingBot-VA** (Li et al., 2026). Figure 3(a) separates its operation into memory writing and future prediction. Memory writing preserves detailed historical features together with compact register representations before the final instruction is known. During prediction, instruction-informed queries select a limited set of historical chunks for detailed access. The register-guided retrieval mechanism in Figure 3(b) combines this selected detail with global registers and recent context to support video and action prediction.

4.1 Backbone and Memory Organization

We retain LingBot-VA's video–action modeling backbone and prediction interface, modifying its access to accumulated history. During execution, current noisy video tokens are first processed with recent context and the task instruction. At a designated intermediate layer r, their queries select historical video chunks. The selected IDs are then reused from layer r onward, and the action-prediction stage reuses the video stage's final selection. Selection may be refreshed at configured steps of video denoising; it is not fixed for an entire episode.

ReWAM retrieves long-range *visual* history and retains local action context. It does not retrieve the corresponding long-range action chunks when selecting a video chunk. This distinction motivates the action-context-controlled LingBot-VA baseline in Section 5.2.

4.2 Register-Based Memory Writing

The memory-writing branch in Figure 3(a) processes observed video through the backbone's VAE encoder and tokenization pipeline. For chunk i, let $X_i \in \mathbb{R}^{P\times d}$ denote the resulting detailed video tokens. We append one learnable register seed $\boldsymbol{\rho} \in \mathbb{R}^{1\times d}$ shared across chunks:

$$\widetilde{X}_i = [X_i; \boldsymbol{\rho}]. \tag{2}$$

The augmented tokens pass through the shared backbone, and we retain detailed-token and register KV states at each layer. Each chunk therefore has one content-dependent register representation without a separate memory encoder. A register query directly attends to its own chunk's video tokens and itself; those video representations may already contain causally accumulated context. Register tokens have no separate decoded output, and action chunks receive no registers.

History writing follows the task-independent boundary of Section 3: the final instruction is withheld while historical features are formed. Detailed history remains cached regardless of which chunks a later query selects. Sparse selection controls prediction-time reads rather than determining which historical chunks are initially stored.

4.3 Instruction-conditioned chunk selection

Figure 3(b) details the register-guided retrieval module in the prediction branch. At layer r, current video-token queries score historical register keys, producing one pooled relevance score per candidate chunk.

Let $q_{mh}^{(r)}$ be the query of ordinary current video token m in head h at layer r, and let $k_{ih}^{\mathrm{reg},(r)}$ be the corresponding register key for historical chunk i. We first average scaled query–key logits over n_h heads, then aggregate over M query tokens using normalized log-mean-exp:

$$\bar{\ell}_{mi} = \frac{1}{n_h} \sum_{h=1}^{n_h} \frac{\langle q_{mh}^{(r)}, k_{ih}^{\mathrm{reg},(r)} \rangle}{\sqrt{d_h}},$$

$$s_i = \tau \left[\log \sum_{m=1}^{M} \exp(\bar{\ell}_{mi}/\tau) - \log M \right]. \tag{3}$$

Here d_h is the head dimension and τ is the query-pooling temperature. These are logit-based scores rather than normalized attention probabilities. The projections and positional transformations follow the retrieval layer's attention configuration. No separate learned retrieval-query token is introduced: the scoring queries are ordinary video-token queries whose representations have interacted with the instruction in preceding blocks.

Given eligible historical chunks $\mathcal{C}$, the detailed read set is

$$\mathcal{S} = \mathrm{TopK}_{i \in \mathcal{C}}(s_i; K). \tag{4}$$

The candidate set contains retained, causally available chunks with both register and detailed features. We exclude the recent set $\mathcal{L}$ from the ranking candidates $\mathcal{C}$, since its detailed tokens are included through the mandatory local context. Holding K fixed limits selected historical detail as the accumulated history grows.

In the illustrative example in Figure 3(b), chunks C_1–C_6 are non-local candidates, while C_7 and C_8 supply mandatory recent context. With $K = 2$, the selector retrieves detailed K/V from C_2 and C_5, while register K/V from all eight chunks remain available. The displayed scores and selections are schematic.

4.4 Prediction with compact, selected, and local memory

Before layer r, ordinary prediction queries access recent visual context and the permitted action context. From layer r onward, the historical video K/V set combines three sources, shown at the bottom of Figure 3(b): registers from all causally available retained chunks, detailed tokens from the selected set $\mathcal{S}$, and detailed tokens from the recent set $\mathcal{L}$. Selected and local detail form the union $\mathcal{S} \cup \mathcal{L}$.

These sources are combined into one K/V set rather than processed by separately normalized attention branches. Together with current-token K/V and permitted local action K/V, they form the visible set $\mathcal{V}^{(\ell)}$ at layer ℓ. The figure expands the historical video-memory component; current-token and local action K/V are omitted from panel (b) for clarity. Prediction uses a single attention operation over the combined set:

$$Y^{(\ell)} = \mathrm{Attn}\Big(Q^{(\ell)}, K_{\mathcal{V}}^{(\ell)}, V_{\mathcal{V}}^{(\ell)}\Big). \tag{5}$$

Local and selected detailed tokens are deduplicated. The same selection IDs index each later layer's own cache, preserving the backbone's layer-specific historical features. Registers participate in actual prediction attention in addition to ranking chunks, providing compact global information even when their associated detailed tokens are not selected.

We retain the backbone's visual and action prediction objectives. Hard top-K membership is discrete, while attention to register KV provides a differentiable prediction path wherever those representations remain connected to the training graph. This does not make the selection operation differentiable or guarantee correct retrieval.

5 EXPERIMENTS

Our evaluation addresses two questions. First, can ReWAM complete memory-dependent manipulation tasks on RMBench and MOSAIC compared with Dense and Local-A? Second, how does its per-generation latency scale with retained history length?

5.1 BENCHMARKS AND EXPERIMENTAL SETUP

We evaluate manipulation success on RMBench (Chen et al., 2026) and MOSAIC using a common LingBot-VA backbone. The reported RMBench results cover seven tasks. For each policy, the RMBench model is jointly trained across tasks for 5,000 optimization steps. For MOSAIC, we evaluate the three- and six-target configurations of Store and Retrieve Blocks, training a separate model for each policy and configuration for 5,000 optimization steps.

Each policy is evaluated on 90 episodes per task. We report task-level success rates and the unweighted mean over the seven reported RMBench tasks. The two MOSAIC configurations are reported separately; we do not report an aggregate over the unevaluated families. Because the benchmarks use different training regimes, cross-benchmark differences are not attributed solely to history length.Training and retrieval settings are provided in Appendix A.1.

5.2 BASELINES

We compare three policies built on LingBot-VA. All policies receive the current inputs and task instruction. **LingBot-VA (Dense)** uses all retained detailed visual and action history. **LingBot-VA (Local-A)** retains dense visual history but restricts past actions to the same local context used by ReWAM. **ReWAM** combines global video registers, instruction-selected video detail, and local visual and action context.

Dense provides a full-history reference. Local-A matches the action context of ReWAM, allowing their comparison to assess the overall visual-memory design. This comparison includes both compact registers and selective detailed reads; it does not isolate either component alone.

5.3 MANIPULATION PERFORMANCE ON RMBENCH AND MOSAIC

Table 1 reports success rates on seven RMBench tasks and the three- and six-target configurations of Store and Retrieve Blocks in MOSAIC.

RMBench. ReWAM achieves a mean success rate of 50.57%, compared with 55.00% for Dense and 44.71% for Local-A. Relative to Dense, ReWAM performs better on three tasks but trails by 4.43 percentage points on average. Under matched local action context, it exceeds Local-A on six of the seven tasks, improving the mean by 5.86 percentage points. The results therefore show a task-dependent trade-off relative to full-history conditioning, while supporting the benefit of the ReWAM memory design over Local-A. Absolute success remains low for all three policies on `observe_and_pickup` and `cover_blocks`, indicating that these tasks remain challenging.

MOSAIC. On the three-target configuration, ReWAM achieves 93% success, compared with 36% for Dense and 67% for Local-A. On the six-target configuration, it achieves 45%, exceeding Dense at 26% but trailing Local-A at 55%. Both policies with local action context outperform Dense on both configurations, suggesting that retaining the full action history is not uniformly beneficial

Table 1: Manipulation success (%) over 90 episodes per policy and task. The RMBench mean equally weights seven tasks. MOSAIC results cover the three- and six-target configurations of Store and Retrieve Blocks. Bold marks the highest rate per row.

Task	Dense	Local-A	ReWAM
RMBench			
put_back_block	**88**	40	68
rearrange_blocks	76	96	**100**
swap_T	**68**	58	49
observe_and_pickup	8	7	**12**
cover_blocks	**12**	6	8
swap_blocks	78	83	**86**
press_button	**56**	24	31
Mean over reported tasks	**55.00**	44.71	50.57
MOSAIC			
Store and Retrieve Blocks (3 targets)	36	67	**93**
Store and Retrieve Blocks (6 targets)	26	**56**	46

in these settings. However, the six-target result also exposes a performance trade-off for selective visual access relative to Local-A.

These comparisons do not isolate history length as the cause of the cross-benchmark differences, since task structure and training regime also differ. Together with the separate latency measurements in Section 5.4, they characterize the performance and computational trade-offs of selective historical access.

5.4 GENERATION LATENCY VERSUS HISTORY LENGTH

We compare the inference latency of Dense, Local-A, and ReWAM as retained history increases from 1 to 256 video chunks.

The current input size, model dimensions, precision, batch size, and denoising schedule are held fixed across history lengths. For ReWAM, the retrieval layer, selected-chunk budget, and local windows are also fixed. Cache capacity is sufficient to retain the requested history without eviction or hidden truncation.

Figure 4 reports directly measured latency for one complete model-side video–action inference cycle using 25 video denoising steps and 50 action denoising steps. We time the full generation routine, including denoising, retrieval, KV gathering, and intermediate cache operations, rather than estimating its latency from isolated denoising steps. Historical cache preparation occurs before timing; environment execution, communication, and image/text encoding are excluded. Measurements use warm-up runs and GPU synchronization. Each policy–history-length setting is measured with three random seeds, using 50 timed repetitions per seed for individual denoising steps and 10 per seed for complete inference cycles.

We additionally measure individual video and action denoising steps to examine the two prediction stages separately (Appendix B). These component measurements supplement the directly timed full-cycle result.

Figure 4 shows that the efficiency advantage of ReWAM increases with retained history length. At 256 video chunks, directly measured full-cycle inference latency is 8.99 s for ReWAM, compared with 27.47 s for Dense and 25.23 s for Local-A, corresponding to speedups of $3.06\times$ and $2.81\times$, respectively. The Local-A comparison shows that restricting action history alone does not account for the observed efficiency benefit; selective visual-memory access provides a further reduction under matched local action context.

As history increases from 1 to 256 chunks, the full-cycle latency of ReWAM rises by only 4.27%, compared with 239.91% for Dense and 211.62% for Local-A. At one chunk, however, ReWAM is 6.67% slower than Dense. Among the tested lengths, it first outperforms both baselines at 16 chunks, with the advantage widening at longer histories. These results demonstrate limited latency growth

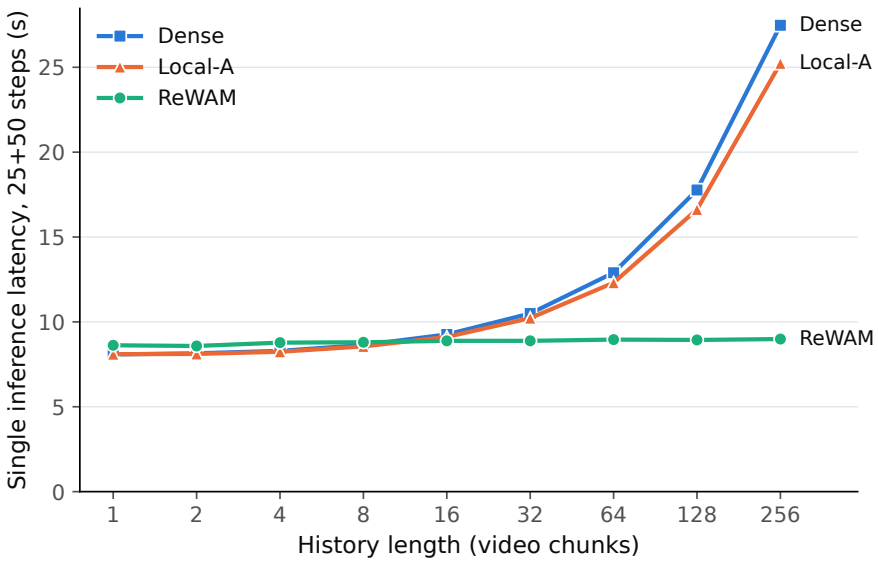


Figure 4: **Inference latency versus history length.** Latency is measured directly over a complete model-side video–action inference cycle using the 25+50-step schedule. ReWAM exhibits limited latency growth over the tested range, while Dense and Local-A become increasingly expensive.

over the evaluated range, rather than history-independent computation: access to global registers still grows with retained history.

6 Limitations

Our manipulation evaluation covers seven RMBench tasks and two configurations of one MOSAIC family. Different training regimes limit cross-benchmark comparisons, and controlled tabletop tasks do not establish reliable recall in unconstrained deployment. Task success combines memory, reasoning, and control, without separately validating selected evidence.

Compact registers can discard detail, and hard selection can miss complementary events. Global-register access and retained-cache storage still grow with history length.

7 Conclusion

We study late-bound manipulation, in which a robot accumulates experience before receiving the instruction that determines which historical evidence is relevant. MOSAIC operationalizes this setting through shared-history instruction branches, while ReWAM combines compact global memory, selected historical detail, and local context.

On the seven evaluated RMBench tasks, ReWAM improves mean success over Local-A while trailing Dense. It outperforms Dense on both evaluated MOSAIC configurations, although Local-A performs better on the six-target variant. Separately, direct timing at 256 historical video chunks shows $3.06\times$ and $2.81\times$ full-cycle speedups over Dense and Local-A. These results demonstrate the efficiency benefits and task-dependent performance trade-offs of selective historical access.

AI Use Statement

Generative AI tools were used to assist with manuscript drafting and revision, literature discovery, feedback on experimental design, and preparation of scientific figures. The authors take responsibility for the final content of this work, including its claims, results, and artifacts.

REFERENCES

Anuva Banwasi, William Muckelroy, III, Priya Sundaresan, Linfeng Zhao, Jeannette Bohg, and Cherie Ho. MessyMem: Learning-from-Doing Memory for Mobile Manipulation. arXiv preprint arXiv:2609.15976, 2026. URL https://arxiv.org/abs/2609.15976.

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. arXiv preprint arXiv:2603.01229, 2026. 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. arXiv preprint arXiv:2502.10550, 2026. URL https://arxiv.org/abs/2502.10550.

Nhat Chung, Taisei Hanyu, Toan Nguyen, Huy Le, Frederick Bumgarner, Duy Minh Ho Nguyen, Khoa Vo, Kashu Yamazaki, Chase Rainwater, Tung Kieu, Anh Nguyen, and Ngan Le. Rethinking Progression of Memory State in Robotic Manipulation: An Object-Centric Perspective. arXiv preprint arXiv:2511.11478, 2025. URL https://arxiv.org/abs/2511.11478.

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. arXiv preprint arXiv:2603.04639, 2026. URL https://arxiv.org/abs/2603.04639.

Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning Universal Policies via Text-Guided Video Generation. arXiv preprint arXiv:2302.00111, 2023. URL https://arxiv.org/abs/2302.00111.

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. arXiv preprint arXiv:2501.18564, 2025. URL https://arxiv.org/abs/2501.18564.

Yihuai Gao, Jeff Jinyun Liu, Shuang Li, and Shuran Song. Gated Memory Policy: In-Context Memorization and Adaptation. arXiv preprint arXiv:2604.18933, 2026. URL https://arxiv.org/abs/2604.18933.

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.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos Policy: Fine-Tuning Video Models for Visuomotor Control and Planning. arXiv preprint arXiv:2601.16163, 2026. URL https://arxiv.org/abs/2601.16163.

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. URL https://arxiv.org/abs/2605.10921.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal World Modeling for Robot Control. arXiv preprint arXiv:2601.21998, 2026. URL https://arxiv.org/abs/2601.21998.

Max Sobol Mark, Jacky Liang, Maria Attarian, Chuyuan Fu, Debidatta Dwibedi, Dhruv Shah, and Aviral Kumar. BPP: Long-Context Robot Imitation Learning by Focusing on Key History Frames. arXiv preprint arXiv:2602.15010, 2026. URL https://arxiv.org/abs/2602.15010.

Soumil Rathi. RoboMME-Interference: Benchmarking Robot Memory Under Interference. arXiv preprint arXiv:2606.22338, 2026. URL https://arxiv.org/abs/2606.22338.

Yichao Shen, Fangyun Wei, Zhiying Du, Yaobo Liang, Yan Lu, Jiaolong Yang, Nanning Zheng, and Baining Guo. VideoVLA: Video Generators Can Be Generalizable Robot Manipulators. arXiv preprint arXiv:2512.06963, 2025. URL https://arxiv.org/abs/2512.06963.

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. arXiv preprint arXiv:2508.19236, 2026. URL https://arxiv.org/abs/2508.19236.

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. MemER: Scaling Up Memory for Robot Control via Experience Retrieval. arXiv preprint arXiv:2510.20328, 2025. URL https://arxiv.org/abs/2510.20328.

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.

Kai Wang, Zhaopeng Gu, Yixiang Chen, Yuan Xu, Qisen Ma, Jiabing Yang, Zhaowen Li, Yan Huang, Liang Wang, and Peng Su. DIM-WAM: World-Action Modeling with Diverse Historical Event Memory. arXiv preprint arXiv:2606.27677, 2026. URL https://arxiv.org/abs/2606.27677.

Ganlin Yang, Zhangzheng Tu, Yuqiang Yang, Sitong Mao, Junyi Dong, Tianxing Chen, Jiaqi Peng, Jing Xiong, Jiafei Cao, Jifeng Dai, Wengang Zhou, Yao Mu, and Tai Wang. EventVLA: Event-Driven Visual Evidence Memory for Long-Horizon Vision-Language-Action Policies. arXiv preprint arXiv:2606.20092, 2026a. URL https://arxiv.org/abs/2606.20092.

Sizhe Yang, Juncheng Mu, Tianming Wei, Chenhao Lu, Xiaofan Li, Linning Xu, Zhengrong Xue, Zhecheng Yuan, Dahua Lin, Jiangmiao Pang, and Huazhe Xu. MemoryWAM: Efficient World Action Modeling with Persistent Memory. arXiv preprint arXiv:2606.20562, 2026b. URL https://arxiv.org/abs/2606.20562.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World Action Models are Zero-shot Policies. arXiv preprint arXiv:2602.15922, 2026. URL https://arxiv.org/abs/2602.15922.

Mingfeng Yuan, Hao Zhang, Mahan Mohammadi, Runhao Li, Jinjun Shan, and Steven L. Waslander. STaR: Scalable Task-Conditioned Retrieval for Long-Horizon Multimodal Robot Memory. arXiv preprint arXiv:2602.09255, 2026a. URL https://arxiv.org/abs/2602.09255.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do World Action Models Need Test-time Future Imagination? arXiv preprint arXiv:2603.16666, 2026b. URL https://arxiv.org/abs/2603.16666.

A Dataset and Evaluation Details

A.1 Training and retrieval configuration

Table 2 summarizes the training and retrieval settings. RMBench policies are jointly trained across tasks, whereas MOSAIC policies are trained separately for each evaluated configuration. Each run uses 5,000 optimization steps.

Table 2: Training and ReWAM retrieval configuration. The batch size is reported per GPU.

Parameter	Value
ReWAM retrieval	
Retrieval layer r	4
Selected-chunk budget K	3
Local window	2 chunks
Query-pooling temperature τ	0.8
Training	
Learning rate	1×10^{-5}
Batch size per GPU	1
Gradient accumulation steps	1
Optimization steps	5,000

A.2 Task families and descriptions

MOSAIC contains seven task families and eight configurations. Store and Retrieve Blocks has three- and six-target variants. Table 3 summarizes their memory demands, using the task names shown in Figure 2.

Store and Retrieve Blocks. Objects are stored under covers at different times, so their identity–location associations are distributed across the history. After the objects are covered, the instruction specifies which object to access. The robot must open the corresponding cover. The three- and six-target configurations vary the number of candidate objects and locations.

Replay Block Trajectory. Red, green, and blue blocks demonstrate routes through a set of positions at different times during the shared history. After the demonstrations, the final instruction specifies which colored block's route the grey block should reproduce. All instruction branches begin from the same execution-start state, as illustrated in Figure 1. The task requires associating the requested demonstrator with its historical trajectory and reproducing the ordered positions using the grey block.

Return to K-th Light. A block moves along a conveyor while a shared lamp activates at selected positions. After the history, the instruction specifies an activation index k, and the robot must return the block to the corresponding position. The conveyor is stopped and the lamp is off during execution. Solving the task requires relating the temporal order of light activations to their spatial locations.

Audit Tote History. Objects enter and leave different containers during the history. The final instruction asks for a container satisfying a combination of conditions about its past contents. The relevant evidence can span multiple events, requiring the robot to combine historical facts rather than rely only on the current container contents.

Calibrate Relay Routes. The history demonstrates two stages of a relay mapping separately. The final instruction specifies a desired output, and the robot must select an input whose composed route reaches that output. The required evidence therefore includes the relevant associations from both stages.

Table 3: Task families and memory demands in MOSAIC. Store and Retrieve Blocks includes three- and six-target configurations.

Task family	Historical information and requested action	Memory demand
Store and Retrieve Blocks	Recall where a requested object was stored and open the corresponding cover.	Identity–location binding
Replay Block Trajectory	Move the grey block along the demonstrated route of the requested colored block.	Ordered segment recall
Return to K-th Light	Return the block to the position associated with the requested kth light activation.	Event order and location
Audit Tote History	Select a container satisfying the requested combination of historical conditions.	Multi-event conjunction
Calibrate Relay Routes	Combine separately demonstrated mappings to activate the requested output.	Relational composition
Scan and Retrieve Parcels	Associate a transient scan identifier with a stored parcel and retrieve the requested parcel.	Transient identity binding
Sort into Hidden Bins	Apply demonstrated category–bin assignments to sort the requested objects.	Historical rule application

Scan and Retrieve Parcels. Parcels pass through a scanning region, where identifiers from the fixed set $\{1, 2, 3\}$ are briefly displayed, and are subsequently stored under covers. The final instruction identifies a parcel to retrieve. The robot must connect its scan identifier with its later storage location, although the identifier is no longer visible at execution time.

Sort into Hidden Bins. Demonstrations establish category-to-bin assignments for the current episode. The robot later sorts the requested objects according to those assignments. The task requires recalling the demonstrated relation between object categories and destination bins.

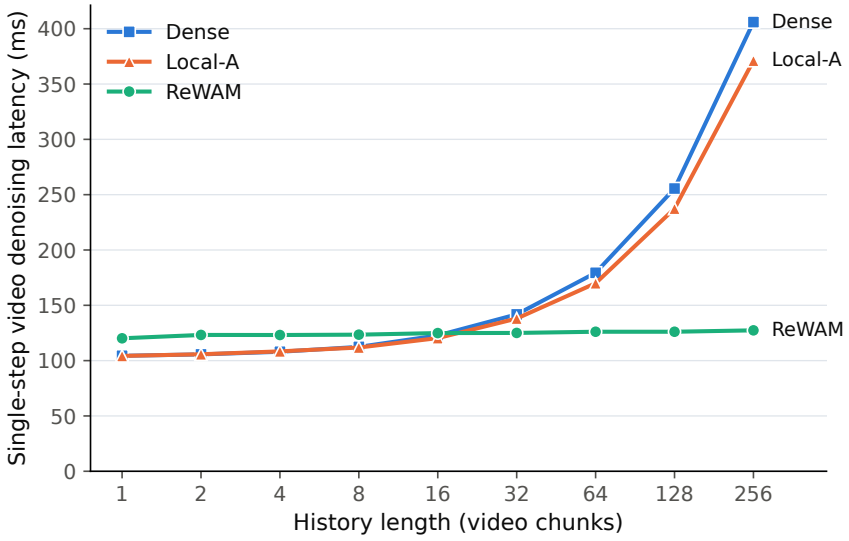
A.3 Dataset and evaluation settings

The evaluation examines manipulation performance and computational efficiency. Manipulation experiments compare Dense, Local-A, and ReWAM on seven RMBench tasks and the three- and six-target Store and Retrieve Blocks configurations of MOSAIC. Latency experiments compare the same three policies as retained history length increases.

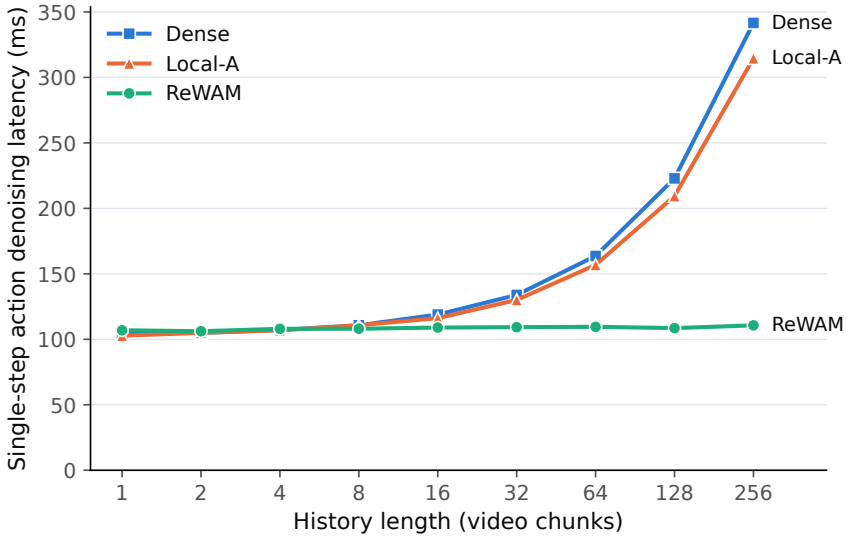
B Latency Breakdown

Figure 5 reports latency for an individual video denoising step and an individual action denoising step under Dense, Local-A, and ReWAM. These measurements complement the directly timed complete inference cycles in Figure 4; the full-cycle measurements are not estimated by summing the isolated step timings.

At 256 historical video chunks, ReWAM reduces video-step latency from 406.00 ms to 127.37 ms relative to Dense ($3.19\times$ speedup), and action-step latency from 341.63 ms to 110.66 ms ($3.09\times$ speedup). Relative to Local-A, whose corresponding latencies are 371.13 ms and 314.61 ms, the speedups are $2.91\times$ and $2.84\times$. Thus, the efficiency benefit appears in both video and action prediction.



(a) Single video denoising step



(b) Single action denoising step

Figure 5: **Per-step denoising latency.** Video and action denoising latency are measured separately in milliseconds. Dense and Local-A retain detailed visual history, whereas ReWAM selectively accesses historical detail. The latency benefit of selective access increases with history length in both prediction stages.