T²MEM: LEARNING TEST-TIME MEMORY FOR ROBOTICS

Anonymous authors
Paper under double-blind review

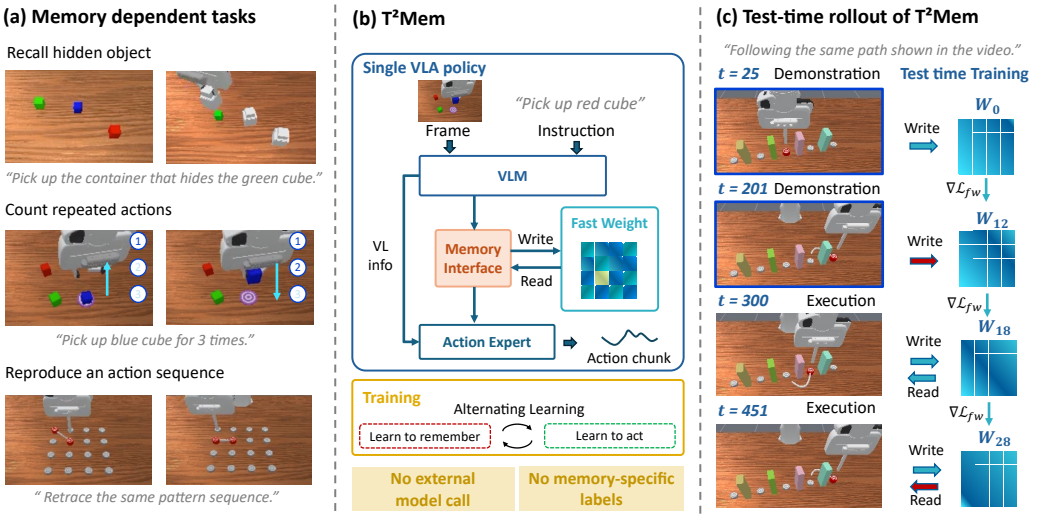


Figure 1: **Test time memory (T²Mem): learning to remember for memory-dependent robot manipulation.** (a) Tasks such as recalling hidden objects, counting repeated actions, and reproducing action sequences require information beyond the current observation. (b) T²Mem integrates fast-weight memory into a single VLA policy through an observation-grounded interface, making vision–language history information available to the action expert. Alternating memory–policy learning develops the ability to retain and use history, without external model calls or memory-specific annotations. (c) During a rollout, the policy uses test-time training (TTT) to encode observation history in fast weights through self-supervised updates. It retrieves this information to guide actions when relevant evidence is no longer visible.

ABSTRACT

Memory-dependent robotic manipulation requires policies to use information that is no longer available in the current observation. Retaining history alone is insufficient: memory must preserve information that supports future actions. We introduce T²MEM, a framework that develops this capability within a pretrained vision-language-action policy, without external reasoning models or memory-specific annotations. T²MEM uses test-time training to encode observation history into compact fast weights through online self-supervised updates, avoiding repeated processing of the full history. An observation-grounded interface extracts vision-language information for memory formation and supplies retrieved context to the action expert. Action supervision shapes what the memory learns to retain and use, while alternating memory–policy learning gives each component a fixed counterpart during optimization. Across 16 RoboMME tasks, T²MEM improves average success from 17.93% to 56.83% over the memory-free base policy and outperforms the recurrent-memory methods reported in the benchmark, while controlled profiling indicates at least 3× inference speedup over explicit methods.

1 INTRODUCTION

Memory is not a verbatim record of the past, nor is it formed in isolation from the functions it serves: past experience is internally represented and later recovered in ways shaped by how it is ultimately used for future decisions (Tulving & Thomson, 1973; Morris et al., 1977; Schacter et al., 1998). For robots, effective memory is therefore neither limited to the current observation nor retaining the

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.

full interaction history in a long context window, but instead lies in forming an internal memory state shaped by the decisions it must support — for example, remembering which object was hidden to select the correct target after occlusion, or the order of a demonstrated sequence to reproduce the same sequence during execution. Memory and action are therefore reciprocal: memory guides future action, while action supervision shapes how memory is formed and used.

Such decision-relevant memory naturally belongs within the decision-making system itself, rather than being constructed as an explicit representation and then passed to the policy through separate storage, retrieval, or reasoning modules. Explicit memory introduces an intermediate bottleneck: information discarded or distorted before action prediction cannot be recovered downstream, and the memory representation is not necessarily shaped by the decisions it must support. This points to an implicit, end-to-end form of memory, where memory formation and use are jointly shaped by action, echoing cognitive accounts in which memory is reconstructive over internal representations (Bartlett, 1932). Test-time training (TTT) offers a natural computational mechanism for this view: instead of explicitly retaining history, past observations can be absorbed into fast weights through online self-supervised updates (Sun et al., 2020; 2024). These online updates can encode history. However, encoding history alone does not ensure that the resulting memory contains the information most useful for action. This motivates an internal memory mechanism whose online updates remain self-supervised, while its formation is shaped by action supervision toward downstream control.

We propose T²Mem, which integrates adaptive **T**est-**T**ime **MEM**ory directly into a pretrained VLA, making memory an intrinsic part of the policy. An *observation-grounded memory interface* between the vision-language backbone and the action expert encodes observation history into fast weights through self-supervised updates and queries this evolving memory to support history-conditioned actions. Our *alternating memory–policy learning* couples memory formation with its use in control: action supervision shapes how memory is formed and used, while alternating optimization gives memory and policy a fixed counterpart when learning to remember and to act, respectively. At deployment, the policy remains fixed while memory adapts from observations alone. The method requires no memory-specific annotations, auxiliary task-state targets, or external model calls, and learns from task demonstrations without an additional broad sequence-pretraining stage.

We evaluate T²MEM on RoboMME, which comprises 16 memory-dependent tasks across four categories (Dai et al., 2026). T²MEM improves average success from 17.93% to 56.83% over the memory-free base policy and outperforms the recurrent-memory methods reported in the benchmark. Memory interventions and efficiency analysis further examine whether the policy uses stored history and at what inference cost. Our central question is whether a given pretrained policy can learn to extract, retain, and use decision-relevant information through action supervision and online observation-based self-supervision, without larger external reasoning models or additional memory annotations. Beyond task success, we aim to make memory an intrinsic capability of a single policy. This approach could extend to more advanced foundation models, providing a foundational memory capability that complements explicit memory and higher-level reasoning for capable and efficient robotic control.

2 Related Work

Memory in Robotics Memory-augmented policies use history to address partial observability. Active Neural SLAM maintains spatial maps (Chaplot et al., 2020), while Scene Memory Transformer attends to stored observation embeddings (Fang et al., 2019). Spatial maps primarily capture geometry; observation banks require managing retention and retrieval costs. For manipulation, SAM2Act+ supports spatial recall through a memory bank (Fang et al., 2025), and MemoryVLA consolidates perceptual and semantic features (Shi et al., 2026). These approaches trade historical detail against storage and attention costs. Recurrent methods instead compress history into persistent states. RoboFlamingo uses a recurrent policy head (Li et al., 2024), while ReMem-VLA combines dual-level recurrent queries with auxiliary past-observation reconstruction (Li et al., 2026). However, reconstructing past observations is not equivalent to retaining decision-relevant information. MemoryBench and RoboMME evaluate whether historical information supports decisions beyond the current observation (Fang et al., 2025; Dai et al., 2026).

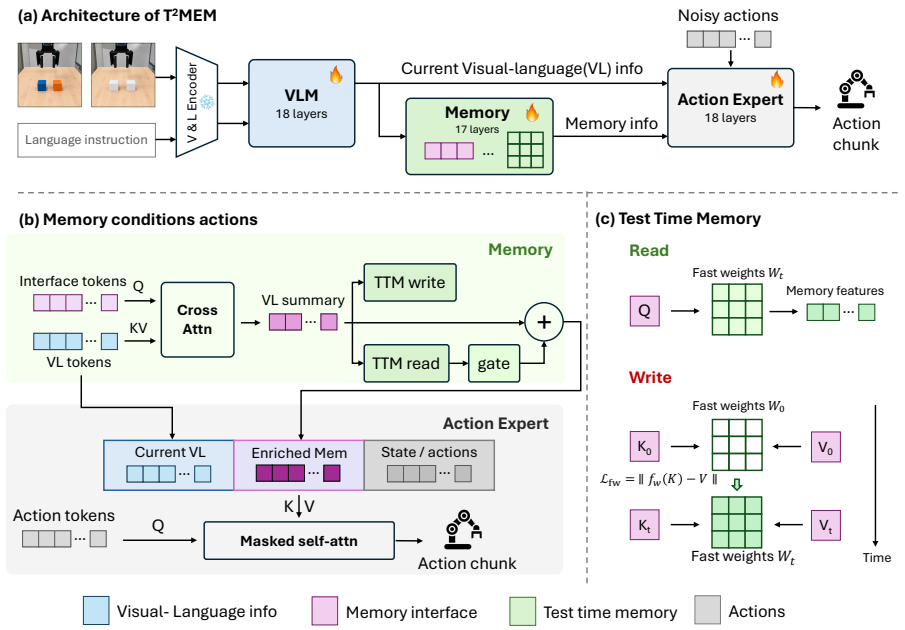


Figure 2: **Architecture of T²Mem.** (a) Memory connects the VLM and AE alongside the direct vision-language pathway. (b) The observation-grounded interface extracts vision-language features, fuses memory readouts, and supplies attention context to the AE. (c) Self-supervised updates encode history in fast weights.

Test-Time Training Fast-weight models store temporary associations in rapidly changing parameters (Ba et al., 2016). TTT adapts models through test-time self-supervision (Sun et al., 2020). Subsequent work introduces feature alignment (Liu et al., 2021), masked reconstruction (Gandelsman et al., 2022), and adaptation over video streams (Wang et al., 2025). TTT layers encode history through online updates to a learned memory model (Sun et al., 2024), while Titans adds surprise-driven updates, momentum, and forgetting (Behrouz et al., 2025). Applying these mechanisms to robotics requires bridging visual observations, semantic representations, and action generation: memory must retain information in a form the policy can use. RoboTTT integrates fast weights into robot policies, emphasizing visuomotor context scaling and in-context adaptation (Jiang et al., 2026). However, longer context and memory-dependent decision making are distinct objectives. Tasks requiring semantic memory depend on selectively retaining task-relevant cues and retrieving them when current observations are insufficient, rather than merely extending the history available to action prediction.

3 T²MEM

T²Mem addresses memory-dependent tasks in which the current observation is insufficient for action selection (Figure 2). It connects a vision-language model (VLM) to an action expert (AE) through a parametric memory pathway. The memory extracts vision-language features stores as semantic history through adaptive self-supervised updates, and provides historical context for action prediction. No memory-specific annotations or auxiliary task-state targets are required. Expert actions supervise the outer training objective; at deployment, memory updates use observations alone.

3.1 PRELIMINARIES

Memory-Dependent Tasks Let o_t denote the robot's visual and proprioceptive observation, ℓ a language instruction, and $h_t = (o_0, a_0, \ldots, a_{t-1}, o_t)$ the interaction history. Define Π_{hist} as the class of history-conditioned policies $\pi_t(a_t \mid h_t, \ell)$ and Π_{current} as the class of observation-conditioned policies $\pi_t(a_t \mid o_t, \ell)$. Under a fixed task distribution and interaction budget, let $J(\pi) = \mathbb{E}_\pi[R(\tau)]$ denote expected task return for trajectory τ, with R optionally defined as the success

indicator. We call a task *memory-dependent* if

$$\sup_{\pi \in \Pi_{\text{hist}}} J(\pi) > \sup_{\pi \in \Pi_{\text{current}}} J(\pi), \tag{1}$$

so that the current observation alone is insufficient for optimal performance. Both policy classes may depend on time t; the distinction concerns access to past observations and actions.

Test-Time Training TTT updates a model at inference using a self-supervised objective constructed from its inputs. In the fast-weight formulation, a neural model f_{W_t} encodes episode history in its parameters W_t (Sun et al., 2024). Given an input representation x_t, slow parameters θ produce queries, keys, and values (q_t, k_t, v_t). A standard associative objective is

$$\mathcal{L}_{\text{mem}}(W; x_t) = \frac{1}{2} \left\| f_W(k_t) - v_t \right\|_2^2, \tag{2}$$

with memory readout and update

$$\begin{aligned} m_t &= f_{W_t}(q_t), \\ W_{t+1} &= W_t - \eta_t \, \nabla_W \mathcal{L}_{\text{mem}}(W; x_t)|_{W=W_t}. \end{aligned} \tag{3}$$

The action policy is conditioned on the readout, $\pi_\theta(a_t \mid o_t, \ell, m_t)$. We adopt a read-before-write convention with inner-loop step size η_t; setting $\eta_t = 0$ skips a write. The slow parameters θ and initialization W_0 are learned through the outer action objective. At deployment, θ remains fixed, while fast weights update without expert action labels and reset to W_0 at each episode boundary.

3.2 Architecture of T²Mem

As shown in Fig. 2(a), T²Mem builds on $\pi_{0.5}$ (Physical Intelligence et al., 2025) and comprises a VLM backbone, a memory module, and an AE. The memory sits between the VLM and AE. Its *memory interface* extracts vision-language features and combines them with retrieved history to condition the AE. This pathway complements the original VLM-to-AE connection, which preserves direct access to the current scene.

We distinguish *slow parameters*, learned across episodes, from *fast state*, updated within an episode. Slow parameters include the policy, interface, memory projections, fusion gates, and fast-weight initialization W_0. They learn how to encode and use history, which responsible for "*how to memorize*" Fast state W_t consists of the current weights and biases of the memory networks and carries episode-specific information, which is "*what to memorize*". Tasks share slow parameters, while each episode starts from W_0 with an independent fast state. During inference, all slow parameters remain fixed, while fast parameters undergo continuous self-supervised updates. During training, all parameters are unfrozen except those of the vision and language encoders.

3.3 Observation-Grounded Memory Interface

The memory interface uses learned query tokens to aggregate VLM features into a compact state summary for memory writes, retrieval, and action conditioning (Fig. 2(b)). This leverages pretrained vision-language representations without updating memory over the full visual token sequence. At depth l, interface representations $E_t^{(l)}$ extract information from vision-language features $Z_t^{(l)}$:

$$U_t^{(l)} = \text{Attn}\Big(Q_{\text{if}}^{(l)}(E_t^{(l)}), K_{\text{VL}}^{(l)}(Z_t^{(l)}), V_{\text{VL}}^{(l)}(Z_t^{(l)})\Big). \tag{4}$$

An attention mask restricts interface queries to valid vision-language keys, excluding action, proprioceptive, and other interface tokens while preserving the AE's original connections. This grounds memory in observations and blocks a direct action-history shortcut that could reduce imitation loss through action extrapolation rather than task-state tracking.

The summary U_t supports both retrieval and writing (Fig. 2(c)). Omitting layer/head indices, normalization, and positional encoding, retrieval uses an observation-conditioned query:

$$Q_t = U_t \theta_q, \qquad R_t = f_{W_t}(Q_t), \tag{5}$$

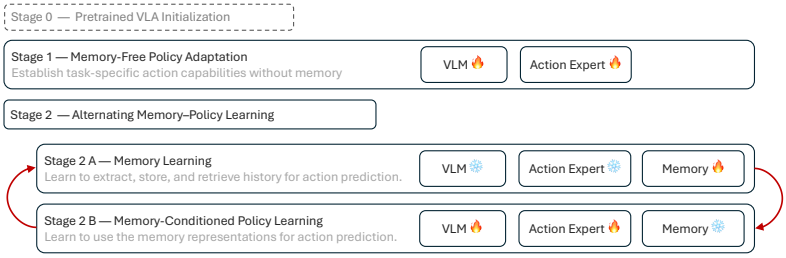


Figure 3: **Alternating memory–policy learning.** Stage 0 initializes a pretrained VLA, and Stage 1 adapts the policy without memory. Stage 2A learns memory with the policy fixed; Stage 2B learns a memory-conditioned policy with the memory mechanism fixed. Red arrows indicate repeated alternation; freezing refers to slow parameters, not episode-specific memory updates.

where f_{W_t} is the fast-weight memory and R_t its history-conditioned readout. For writing, dedicated projections form key–value associations from the same summary:

$$K_t = U_t\theta_k, \qquad V_t = U_t\theta_v. \tag{6}$$

At scheduled write steps, a self-supervised update encodes these associations into the fast weights:

$$\begin{aligned} \mathcal{L}_{\text{mem},t} &= \text{mean}\left[(f_{W_t}(K_t) - V_t)^2\right], \\ W_{t+1} &= W_t - \eta_t^{\text{eff}} \nabla_{W_t} \mathcal{L}_{\text{mem},t}, \end{aligned} \tag{7}$$

where $\eta_t^{\text{eff}} = \eta_t \beta_t$ includes adaptive scaling (Sec. 3.5). Reads precede writes, so new observations affect subsequent decisions. Memory projections θ_q, θ_k, and θ_v are separate from the interface attention projections and learned through the outer action objective; inner updates modify only episode-specific fast weights.

A channel-wise gate fuses retrieved history with the current vision-language summary:

$$\widetilde{U}_t^{(l)} = U_t^{(l)} + \tanh(\alpha^{(l)}) \odot R_t^{(l)}. \tag{8}$$

Residual and feed-forward transformations then produce $C_t^{(l)}$. Following RoboTTT (Jiang et al., 2026), the gate initially limits memory's contribution to protect pretrained visuomotor capabilities; action supervision learns its channel-wise fusion strengths.

The enriched interface supplies keys and values to subsequent action-token attention, alongside current vision-language features, proprioception, and action context. An interface update is consumed after its layer, not by that layer's completed attention computation. The AE thus combines VLM perception and retrieved history to predict actions.

3.4 Alternating Memory–Policy Learning

Self-supervised association learning does not by itself ensure decision-relevant memory, and the policy must learn to use the new historical representations. In joint training, memory updates change the context presented to the policy, while policy updates change the action gradients that guide memory learning. The two modules may therefore continually adapt to each other's changing representations, making a stable memory-to-action mapping harder to learn. We address this coupling by alternating their updates, holding one slow-parameter group fixed while optimizing the other. Figure 3 outlines the training stages, which is, **learning to remember and learning to act are separate.**

Initialization and memory-free adaptation. We initialize from pretrained $\pi_{0.5}$ (Stage 0) and adapt the policy without memory (Stage 1). This establishes task-specific visuomotor skills before introducing the memory pathway.

Stage 2A: memory learning. We fix the VLM/AE and train the memory mechanism to extract, store, and retrieve history useful for action prediction. Expert action supervision passes through the frozen AE to shape both current retrieval and earlier memory writes. The fixed policy provides a stable decision-making counterpart, encouraging memory representations that serve its control needs.

Stage 2B: memory-conditioned policy learning. We fix the memory mechanism and adapt the policy to combine current observations with retrieved history. This phase adjusts both the vision-language features supplied to memory and their use by the AE. Only memory slow parameters are frozen: episode-specific fast states continue to accumulate observations through online updates.

Alternating schedule. We repeat these phases so that memory adapts to the current policy, the policy learns to use the resulting representations, and subsequent memory learning responds to the updated policy. This repeated adaptation differs from training memory once and then fitting a policy to it. Both phases use expert action supervision without memory-content or task-progress labels. The action objective, parameter groups, sequence supervision, and training schedule are detailed in Appendix B.

3.5 Adaptive Self-Supervised Memory

Fast memory encodes history in an online-updated nonlinear mapping (Fig. 2(c)). The observation-grounded interface provides queries, keys, and values for retrieval and self-supervised association learning. This update requires no memory-content labels. The initialization, projections, and step-size parameters are learned through outer action supervision and fixed at deployment.

To limit repeated reinforcement from correlated observations, we adapt write strength using the alignment between the descent direction and $W_t - W_0$, together with the normalized reconstruction residual. These signals determine a scale $\beta_t \in [0.1, 1]$ that attenuates aligned updates while preserving stronger updates for less aligned, poorly reconstructed inputs:

$$W_{t+1} = W_t - w_t \, \eta_t \, \beta_t \; \nabla_W \mathcal{L}_{\mathrm{mem},t}(W)|_{W=W_t} \,, \tag{9}$$

where w_t is the binary write mask and η_t is a curvature-calibrated step size with a learned positive multiplier. The non-adaptive control fixes $\beta_t = 1$ without changing the step-size rule.

Reads precede writes, so each update affects only subsequent predictions. The fixed-size fast state avoids a growing history buffer. Architecture, scaling, and gradient details are provided in Appendix A.1; training and inference procedures appear in Appendix B.

4 Experiments

Our experiments address four questions: *(i) Can T²Mem improve performance on memory-dependent tasks? (ii) Which architectural and learning components contribute to its memory capability? (iii) Does the policy rely on the content of its online memory? (iv) How efficiently can the architecture perform inference?* We first describe the evaluation setup, then examine task performance, component ablations, online memory, and inference efficiency.

4.1 Experimental Setup

Policy training. We initialize the policies from pretrained $\pi_{0.5}$ (Physical Intelligence et al., 2025) and follow the training procedure in Sec. 3.4. We train on RoboMME data with 16 tasks. Memory-free adaptation first establishes task-specific visuomotor skills. We then alternate memory learning and memory-conditioned policy learning, using imitation supervision at valid execution frames in both phases. Each trajectory starts from the learned fast-weight initialization, and its visual history updates memory independently of other trajectories. Implementation details and the reference training recipe are given in Appendix B. The training takes 32 H200 for 48 hours.

Benchmark and evaluation. RoboMME contains 16 manipulation tasks in four suites: Counting, Permanence, Reference, and Imitation (Dai et al., 2026). These probe temporal, spatial, object-centric, and procedural memory. Our standard evaluation uses the complete 50-episode test set with three evaluation seeds. The policy predicts 64 actions and executes 32 before replanning. The ablation and counterfactual experiments in Secs. 4.3 and 4.4 use task-specific checkpoints.

Baselines. All baseline scores in Table 1 are quoted directly from the original RoboMME paper (Dai et al., 2026). They include symbolic memory (SimpleSG and GroundSG), perceptual memory (TokenDrop and FrameSamp), and recurrent memory (TTT and RMT). Additional references are $\pi_{0.5}$, $\pi_{0.5}$ with past actions, SAM2Act+, and MemER (Dai et al., 2026; Fang et al., 2025; Sridhar et al.,

Method		Counting				Permanence				Reference				Imitation				AVG
		Bin Fill	Pick Xtimes	Swing Xtimes	Stop Cube	Video Umsk	Button Umsk	Video UmskS	Button UmskS	Pick HighL	Video Repick	Video PlcBtn	Video PlcOrd	Move Cube	Insert Peg	Pattern Lock	Route Stick	
HUMAN PERFORMANCE		96.00	100.0	80.00	78.00	90.00	92.00	92.00	90.00	92.00	92.00	98.00	90.00	90.00	98.00	84.00	86.00	90.50
MME-VLA w/ Symbolic Memory																		
SIMPLESG	Oracle	85.78	99.78	100.0	44.67	33.11	22.00	15.56	15.56	44.00	27.78	31.33	26.00	87.33	10.00	95.33	55.11	49.58
GROUNDSG	Oracle	85.78	100.0	100.0	49.67	98.78	95.00	99.22	80.22	83.33	97.33	100.0	100.0	87.78	15.56	97.00	55.56	84.08
SIMPLESG	Gemini	46.00	63.00	45.00	2.00	29.00	9.00	14.00	2.00	20.00	15.00	26.00	29.00	61.00	4.00	7.00	0.00	23.25
	QwenVL	**77.56**	**95.33**	5.11	0.44	34.22	19.33	15.33	9.56	17.11	**25.33**	33.33	25.11	82.00	3.78	12.67	7.78	29.00
GROUNDSG	Gemini	26.00	18.00	4.00	3.00	36.00	14.00	13.00	0.00	9.00	17.00	12.00	7.00	17.00	0.00	7.00	2.00	11.56
	QwenVL	52.00	**92.67**	7.33	0.00	**88.67**	24.00	**30.67**	14.00	15.11	**25.33**	**54.00**	**31.78**	71.56	3.33	6.67	6.00	32.70
MME-VLA w/ Perceptual Memory																		
TOKENDROP	Context	48.67	85.11	**94.67**	3.11	33.78	31.56	26.22	16.00	20.67	17.78	31.11	25.33	81.33	4.00	12.67	20.00	34.50
	Modul	34.44	83.56	86.00	5.33	28.22	29.33	28.44	**21.33**	21.33	22.00	**59.56**	**36.00**	62.00	**7.11**	**32.44**	**51.56**	38.04
	Expert	54.22	87.56	**91.78**	4.22	26.67	30.44	18.44	18.89	19.33	20.89	36.44	24.67	**87.56**	2.22	16.22	18.22	34.86
FRAMESAMP	Context	41.22	72.00	73.67	13.67	26.89	30.22	20.89	15.22	17.67	15.22	30.00	20.89	77.22	1.22	15.22	19.67	30.68
	Modul	39.56	87.33	**92.00**	**42.00**	32.67	25.11	24.44	18.22	**22.89**	**30.44**	**60.00**	**32.00**	77.78	**7.56**	**53.56**	**66.67**	**44.51**
	Expert	**57.33**	86.22	**94.67**	**28.89**	31.78	25.78	22.89	**20.22**	19.11	23.11	30.00	24.22	**83.11**	2.00	13.56	17.11	36.25
MME-VLA w/ Recurrent Memory																		
TTT	Context	35.56	62.89	42.44	3.33	29.78	22.89	18.44	14.44	20.44	13.11	34.22	20.22	32.44	1.11	1.56	3.56	22.28
	Modul	34.22	65.11	36.67	2.11	27.22	22.11	25.22	14.11	14.56	12.11	32.67	22.33	31.22	1.11	3.56	7.00	21.96
	Expert	34.89	63.78	41.33	4.00	31.78	22.44	19.56	18.00	12.22	9.56	34.00	22.89	33.56	0.89	3.11	5.56	22.35
RMT	Context	32.44	56.89	33.56	5.78	31.33	10.89	17.33	2.00	14.00	3.78	32.00	29.11	25.78	2.00	5.56	8.89	19.46
	Modul	33.33	60.78	37.78	4.67	31.11	11.78	17.78	2.44	17.11	4.22	32.00	31.11	24.67	2.21	3.78	8.00	20.17
	Expert	35.78	60.22	36.00	5.56	28.00	17.11	15.78	2.00	11.78	0.22	24.22	22.67	20.00	1.89	4.22	4.89	18.15
Other Methods																		
$\pi_{0.5}$		30.00	42.89	35.56	6.67	20.44	22.22	18.67	6.67	11.33	0.44	31.11	25.78	26.00	1.56	2.89	4.67	17.93
$\pi_{0.5}$ w/ past actions		26.67	58.33	26.67	4.67	30.67	23.67	20.67	16.00	12.33	8.67	24.00	18.67	34.00	1.00	4.00	5.67	19.73
SAM2Act+		40.00	76.00	25.33	0.00	27.33	**32.00**	18.00	**26.67**	17.33	5.33	24.67	20.00	29.33	0.00	0.00	0.00	21.37
MemER		56.67	79.33	59.33	0.00	**81.33**	**72.00**	**38.00**	**21.33**	**70.67**	**25.33**	30.00	26.00	**82.67**	6.67	16.67	12.00	**42.38**
Stage 1(no memory)		26.67	34.00	31.33	2.00	27.33	6.00	18.00	2.00	11.33	1.33	26.67	27.33	38.00	1.33	3.33	5.33	16.38
Ours		**60.67**	**91.33**	90.67	**70.00**	**88.00**	**59.33**	**29.33**	20.00	**50.67**	**38.67**	35.33	30.00	80.67	**36.00**	**50.00**	**78.67**	**56.83**

Table 1: Evaluation success rates (%) on the 16 RoboMME tasks. **Red** denotes the highest score, **blue** the second-best, and **green** the third-best among non-privileged methods. Our method ranks among the top three on most tasks.

2026). Human and privileged Oracle results are reference points rather than deployable competitors. Baseline provenance and differences in information access are distinguished in Appendix C.

4.2 PERFORMANCE ON MEMORY-DEPENDENT TASKS

We first examine whether T²Mem enables effective control on memory-dependent tasks. Table 1 summarizes the RoboMME evaluation (including no memory baseline). Among methods without privileged information, T²Mem ranks within the top three on most reported tasks, with the highest success rates on VideoRepick, InsertPeg, and RouteStick. It also substantially improves over the memory-free $\pi_{0.5}$ and the evaluated recurrent-memory baselines.

Performance is strongest on tasks that primarily require direct retrieval of an earlier cue or tracking repeated events, such as VideoUnmask and StopCube. In contrast, success remains lower on VideoUnmaskSwap and ButtonUnmaskSwap, where the policy must track swaps and update object–location associations rather than simply recall a stored binding. This contrast suggests a distinction between retaining information and reasoning over it. As a single-model approach, T²Mem relies on the underlying $\pi_{0.5}$ policy for visual feature extraction and temporal reasoning, without an external reasoning model. Its memory supplies historical evidence but does not, by itself, confer the ability to infer how that evidence changes through subsequent events. The weaker swap performance may therefore reflect limitations in the base policy($\pi_{0.5}$)'s temporal reasoning, which improved memory retention alone cannot resolve.

Low-level control imposes a separate limitation on tasks such as InsertPeg. Our qualitative observations on InsertPeg indicate that the policy can identify and approach the intended target yet fail during the final insertion. Thus, terminal success reflects both memory-dependent decision-making and execution precision. The strong relative improvement on InsertPeg, despite its modest absolute success rate, highlights the importance of distinguishing these failure sources.

4.3 ABLATION STUDIES

We examine three components of T²Mem: the memory interface, alternating memory–policy learning, and adaptive memory writing. Two tasks are used to do the experiments: VideoUnmask (Appendix Figure 8)and Movecube(Appendix Figure 9). The model of the ablation experiment was

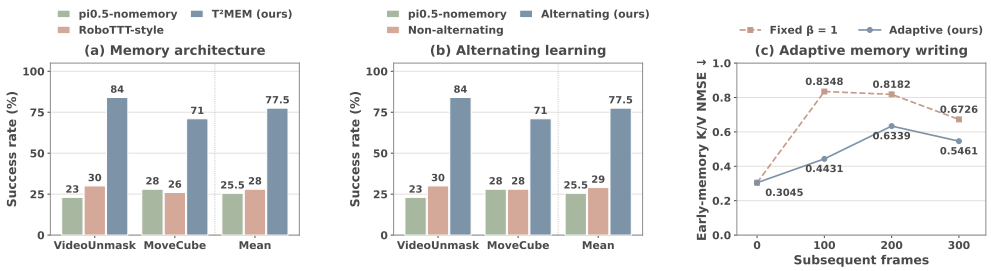


Figure 4: **Ablation studies of T²Mem.** (a) Memory architecture and (b) training schedule: success rates on VideoUnmask and MoveCube, with a memory-free $\pi_{0.5}$ reference. Mean denotes the average across the two tasks. (c) Adaptive writing: reconstruction NMSE of early-memory K/V bindings over subsequent observations, averaged over 20 VideoUnmask Hard episodes. Lower is better; the label $\beta = 1$ denotes $\beta_t = 1$ at every write, disabling adaptive scaling but retaining step-size calibration. Training configurations and comparison scope are detailed in Appendix D.

trained independently on each individual task. Both variants use identical single-task data, batch size, and initialization.

Memory architecture. We compare our architecture with a RoboTTT integration Jiang et al. (2026) on VideoUnmask and MoveCube. The latter places memory within the action expert and derives memory reads and writes from action-expert representations. Figure 4(a) compares these architectures with a memory-free policy. T²Mem achieves 84% and 71% success on the two tasks, compared with 30% and 26% for the AE-side implementation. Appendix D describes the training configurations. This suggests that, unlike retaining past actions for long-context processing, memory-dependent tasks require semantic features extracted from the VLM. T²MEM supports this through its observation-grounded memory interface.

Alternating memory–policy learning. We evaluate alternating and non-alternating optimization on VideoUnmask and MoveCube. Non-alternating training updates memory and policy parameters jointly. Alternating optimization repeatedly adapts memory representations to the policy and the policy to those representations. Figure 4(b) reports joint-training results. In same training budget, non-alternating training shows results similar to no-memory baseline.

Adaptive memory writing. We compare adaptive writing with a $\beta_t = 1$ control on 20 long VideoUnmask Hard episodes, keeping the architecture, write cadence, and step-size normalization unchanged. Starting from shared visible-demonstration memory, we replay 300 subsequent real frames and measure reconstruction error on the initial K/V bindings. Figure 4(c) shows 18.8%–46.9% lower error with adaptive writing, consistently across all 20 episodes. This supports reduced interference with earlier memory bindings under real-trajectory replay.

4.4 Contribution of Online Memory

We intervene at inference time to test whether task success depends on online writes, stored content, and observation timing. These diagnostics use fixed task-specific checkpoints for VideoUnmask, MoveCube, and SwingXtimes, with intervention-specific write schedules detailed in Appendix D.

Disabling online writes. We disable episode-local fast-weight updates while retaining memory reads and the learned initialization. Each condition covers all 50 test episodes and three seeds. Success drops by 82.67 percentage points on VideoUnmask, 64.67 on SwingXtimes, and 41.33 on MoveCube (Figure 5a). The fixed policy and initial memory alone therefore cannot sustain normal performance, demonstrating the importance of online updates, instead learned a genearal solution to all tasks.

Replacing memory content. We construct 50 counterfactual pairs with conflicting targets from VideoUnmask. We reconstruct each target episode, retaining its instruction, environment goal, and diffusion seed, while supplying correct demonstration memory, conflicting donor memory, or empty memory (Appendix D and Figure 11). Across both pairing directions and three seeds, these conditions yield 300/300, 1/300, and 101/300 successes, respectively (Figure 5b). Conflicting memory is more damaging than empty memory, supporting content-specific use of history. Empty-memory per-

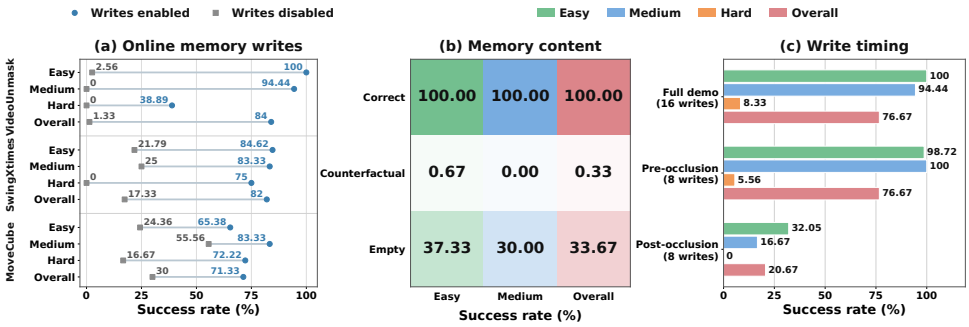


Figure 5: **Online memory supports history-dependent decisions.** (a) Disabling fast-weight writes reduces success across three tasks. (b) Correct, conflicting, and empty memories produce markedly different outcomes on VideoUnmask. (c) Pre-occlusion writes match full-demonstration performance with half the updates, whereas post-occlusion writes perform substantially worse. Success rates (%) are pooled over three seeds under each panel's evaluation protocol.

formance is close to the one-third chance level for three candidates, which is close random selection in no memory policy.

Localizing useful writes. We restrict VideoUnmask writes to different demonstration windows, using stride 4 to provide sufficient updates within each window. The 66-frame prefix contains 16 write opportunities; an image audit identifies the dominant occlusion transition at frame 32 in all eight inspected trajectories. We compare all 16 writes with eight pre-occlusion or eight post-occlusion writes. Under equal write budgets, pre-occlusion writing achieves 76.67%, versus 20.67% after occlusion, and matches full-demonstration performance (Figure 5c). These results localize decision-relevant evidence to the interval in which the target remains visible.

4.5 Inference Efficiency

We profile T²Mem, FrameSamp+Modul, and MemER on RTX A5000 GPUs at batch size one to test the efficiency. Timing uses device synchronization after warm-up, excluding initialization, compilation, communication, and environment execution.As shown in Figure 6, our foreground computation is approximately 3.0× faster than FrameSamp+Modul and 69.2× faster than MemER. Profiling boundaries and workload configurations are detailed in Appendix E.These savings reflect a single-model design that stores and retrieves history in latent space, without external autoregressive reasoning. Its compact parametric memory supports low-latency decisions from historical cues, facilitating fast closed-loop operation on memory-dependent tasks.

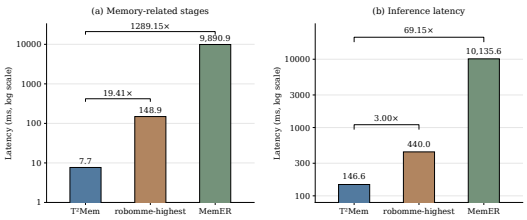


Figure 6: **Controlled-workload inference efficiency.** Left: latency introduced by memory only. Right: end to end latency for an inference.

5 Conclusion

We presented T²Mem, a framework that makes memory an internal capability of a robot policy. An observation-grounded interface connects vision-language perception to fast-weight memory, whose online self-supervised updates retain history for action prediction. Alternating memory–policy learning decouples learning to remember with learning to use memory through expert action supervision, without memory-specific annotations or external model calls. Experiments on the 16 RoboMME tasks demonstrate improved memory-dependent manipulation, while interventions show that decisions depend on both online updates and the information stored. These findings support learning memory formation and use within a single policy as a practical direction for history-dependent robotic control.The remaining challenges lie in the foundation model's intrinsic capabilities for cross-temporal reasoning and information extraction.

AI USE STATEMENT

We used AI-assisted tools throughout the research workflow. These tools assisted with code writing, editing, refactoring, and debugging; infrastructure setup for training and evaluation; job monitoring and troubleshooting; data processing, experimental-result analysis, and visualization; and manuscript drafting, language polishing, and LaTeX formatting. AI assistance also supported literature organization and consistency checks across the manuscript, figures, and tables. The authors retain full responsibility for the research design, implementation, data integrity, experimental results, references, and all claims in the final manuscript.

REFERENCES

Jimmy Ba, Geoffrey E. Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. In *Advances in Neural Information Processing Systems*, volume 29, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/9f44e956e3a2b7b5598c625fcc802c36-Abstract.html. 3

Frederic C. Bartlett. *Remembering: A Study in Experimental and Social Psychology*. Cambridge University Press, Cambridge, 1932. 2

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2501.00663. 3

Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using Active Neural SLAM. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/2004.05155. 2

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. arXiv:2603.04639. Accepted to ICML 2026. 2, 6

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 *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 15925–15942. PMLR, 2025. URL https://proceedings.mlr.press/v267/fang25c.html. 2, 6

Kuan Fang, Alexander Toshev, Li Fei-Fei, and Silvio Savarese. Scene Memory Transformer for embodied agents in long-horizon tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 538–547, 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/html/Fang_Scene_Memory_Transformer_for_Embodied_Agents_in_Long-Horizon_Tasks_CVPR_2019_paper.html. 2

Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei Efros. Test-time training with masked autoencoders. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://papers.neurips.cc/paper_files/paper/2022/hash/bcdec1c2d60f94a93b6e36f937aa0530-Abstract-Conference.html. 3

Yunfan Jiang, Yevgen Chebotar, Ruijie Zheng, Fengyuan Hu, Yunhao Ge, Jimmy Wu, Tianyuan Dai, Scott Reed, Li Fei-Fei, Yuke Zhu, and Linxi "Jim" Fan. RoboTTT: Context scaling for robot policies, 2026. URL https://arxiv.org/abs/2607.15275. arXiv:2607.15275. 3, 5, 8, 15

Hang Li, Fengyi Shen, Dong Chen, Liudi Yang, Xudong Wang, Jinkui Shi, Zhenshan Bing, Ziyuan Liu, and Alois Knoll. ReMem-VLA: Empowering vision-language-action model with memory via dual-level recurrent queries, 2026. URL https://arxiv.org/abs/2603.12942. arXiv:2603.12942. 2

Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, Hang Li, and Tao Kong. Vision-language foundation models as effective robot imitators. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/71639c317fb0bf398835627b4418693e-Abstract-Conference.html. 2

Yuejiang Liu, Parth Kothari, Bastien Germain van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. TTT++: When does self-supervised test-time training fail or thrive? In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://papers.nips.cc/paper/2021/hash/b618c3210e934362ac261db280128c22-Abstract.html. 3

C. Donald Morris, John D. Bransford, and Jeffery J. Franks. Levels of processing versus transfer appropriate processing. *Journal of Verbal Learning and Verbal Behavior*, 16(5):519–533, 1977. doi: 10.1016/S0022-5371(77)80016-9. 1

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, 2025. URL https://arxiv.org/abs/2504.16054. arXiv:2504.16054. 4, 6

Daniel L. Schacter, Kenneth A. Norman, and Wilma Koutstaal. The cognitive neuroscience of constructive memory. *Annual Review of Psychology*, 49:289–318, 1998. doi: 10.1146/annurev.psych.49.1.289. 1

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. MemoryVLA: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2508.19236. 2

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. Scaling up memory for robotic control via experience retrieval. In *International Conference on Learning Representations*, volume 2026, pp. 97142–97166, 2026. 6

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-Time Training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9229–9248. PMLR, 2020. URL https://proceedings.mlr.press/v119/sun20b.html. 2, 3

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (Learn at Test Time): RNNs with expressive hidden states, 2024. URL https://arxiv.org/abs/2407.04620v1. arXiv:2407.04620v1. 2, 3, 4

Endel Tulving and Donald M. Thomson. Encoding specificity and retrieval processes in episodic memory. *Psychological Review*, 80(5):352–373, 1973. doi: 10.1037/h0020071. 1

Renhao Wang, Yu Sun, Arnuv Tandon, Yossi Gandelsman, Xinlei Chen, Alexei A. Efros, and Xiaolong Wang. Test-time training on video streams. *Journal of Machine Learning Research*, 26(9):1–29, 2025. URL https://jmlr.org/papers/v26/24-0439.html. 3