TEMPORAL CONTRASTIVE TEST-TIME TRAINING FOR ROBOTIC MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Robot memory faces two directions in time: the past determines what can be known now, while the future reveals what was worth knowing. Test-time training (TTT) gives this asymmetry an optimization form: the inner objective determines what writes into memory, while outer task losses determine which updates are useful. However, common TTT objectives derive self-supervision primarily from the current observation, providing no direct objective for modeling relations across observed history. We introduce **T**emporal **C**ontrastive **T**est-**T**ime **T**raining (**TC–TTT**), a contrastive inner objective that augments reconstruction by distinguishing the current representation from historical alternatives conditioned on an earlier observation. We further study how to select historical observations for temporal comparison, examining how sampling stride and candidate count affect policy performance. In a simplified model, TC–TTT adds a velocity-like term to the reconstruction gradient, making first-order temporal change explicit. Across nine RMBench tasks, TC–TTT improves mean success from 31.8% to 51.8% over reconstruction-based TTT under the same 77M-parameter Diffusion Policy and action-gradient window. These results identify the inner objective as an important design axis for online robot memory and show that historical contrast can substantially improve control without extending the action-gradient horizon.

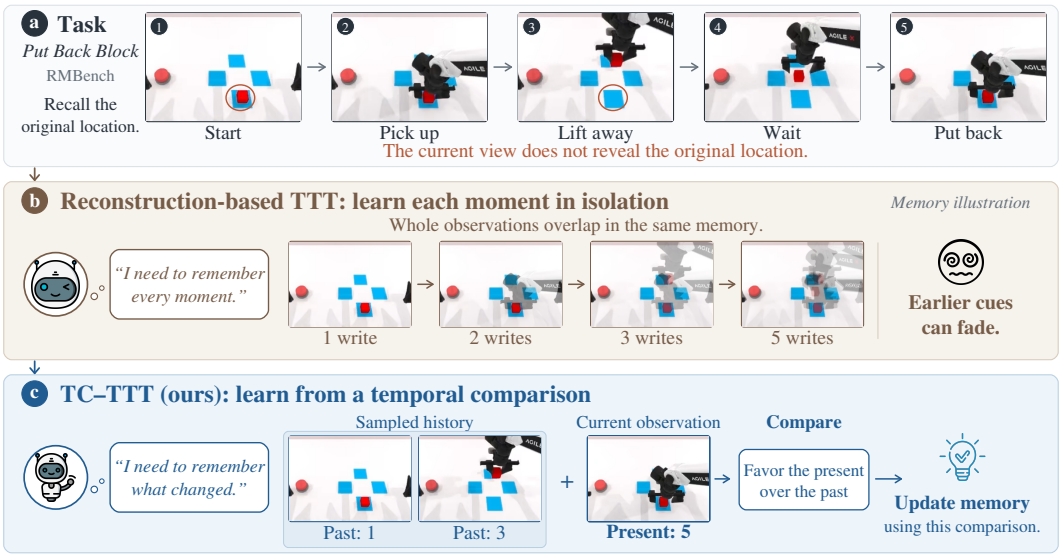


Figure 1: **What the memory is asked to remember.** (*a*) Put Back Block requires remembering the block's original location. (*b*) Reconstruction-based TTT supervises each moment in isolation; overlapping observations illustrate how earlier cues can fade. (*c*) TC–TTT compares the present with sampled history to supervise memory updates. Frames from Chen et al. (2026); memory panels in (*b*) are conceptual illustrations.

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.

1 INTRODUCTION

Robot manipulation often requires knowing not only what is currently visible, but also what happened earlier. Returning an object to its original container requires remembering where it was taken from; continuing a multi-stage procedure requires knowing which steps are already complete; pressing a button the right number of times requires counting the presses already made. In such tasks the current observation alone does not determine the next action, so a policy has to retain information from earlier ones (Kaelbling et al., 1998; Chen et al., 2026; Dai et al., 2026; Lei et al., 2026).

Existing work supplies this information in two broad ways. Some approaches provide the policy with explicit historical context, such as selected past observations or summaries of earlier events (Torne et al., 2026; Mark et al., 2026). Others keep a running memory, updated as each new observation arrives, whose size and cost stay fixed however long the task runs (Gui et al., 2026; Yoo et al., 2026).

We work in that second setting, and with the line of work there that writes the update as a step of optimization: at every observation the memory takes one small training step on an objective the designer chooses, and the same objective runs during policy training and at deployment. This is test-time training (TTT) (Sun et al., 2025; Jiang et al., 2026b; Feng et al., 2026b). It makes what the memory should learn from each observation an explicit choice, one that can be changed without touching the rest of the policy. A common choice is same-observation reconstruction: one observation supplies both a question and an answer about itself, and the memory is trained so that the question retrieves the answer (Sun et al., 2025; Zhang et al., 2025; Behrouz et al., 2025a; von Oswald et al., 2026). Several methods repeat this for many past observations at a time, over a recent window or over the whole episode. Doing so gives the memory more moments to remember, but each reconstruction target remains paired with an input from the same moment, rather than explicitly asking an earlier input to predict a later target. Whether the scene has changed since a moment ago is exactly what the tasks above depend on, and it is something the memory could work out, since those earlier moments have already been observed by the time the update is made. **Can a memory be trained to remember what changed, rather than to remember every moment on its own?**

Our answer is to make the memory's task a comparison. Learning a representation by scoring one outcome against alternatives is standard practice elsewhere (van den Oord et al., 2018), and sensory systems are often described in similar terms, encoding a signal relative to its own context so that what recurs is suppressed and what changes is carried forward (Barlow, 1961; Srinivasan et al., 1982; Rao & Ballard, 1999). We introduce **Temporal Contrastive Test-Time Training (TC–TTT)**, which puts that comparison inside the memory update (Figure 1). The memory is asked to predict the current observation from an earlier one, and to do so against a set of sampled earlier observations that it must not confuse the current one with. Nothing in this task needs to be waited for, so it runs during training and at deployment alike, and it stays one comparison however long the episode grows. Section 3 defines this objective for a differentiable memory model and uses a simplified local analysis to explain the history-dependent learning signal it introduces. Appendix A provides the derivations.

We evaluate TC–TTT on nine RMBench tasks using task-specific policies that augment the RMBench Diffusion Policy backbone with a TTT module. With action gradients reaching no further than the current memory update, TC–TTT raises mean success from 31.7% for reconstruction to 51.8%. History sampling also affects policy performance: in our exploratory scans, increasing the sampling stride at a fixed candidate count improves mean success rates, whereas increasing the candidate count does not consistently improve performance. Our contributions are threefold:

1. **A new inner objective with an interpretable update.** We introduce TC–TTT, a historical-comparison objective for online memory, derive its closed-form update for a matrix memory, and establish a local velocity-like approximation in learned feature space. Write-replacement experiments support this approximation at $s = 1$ (Sections 3 and 4.3).
2. **Substantial gains without longer-horizon action gradients.** Across nine RMBench tasks, TC–TTT raises the mean success from 31.8% to 51.8% using a one-update action-gradient window ($G = 1$), and outperforms reconstruction-only TTT trained with a sixteen-update window ($G = 16$) (Section 4.1).
3. **A study of historical supervision.** We characterize how history stride and candidate count shape the memory update and policy performance. Increasing stride improves mean success in both fixed-K scans, while candidate-count scans show a non-monotonic pattern. (Sections 3.3 and 4.2).

2 PRELIMINARIES: TEST-TIME TRAINING

Test-time training (TTT) represents memory as a model that learns from the observations it receives (Sun et al., 2025). A self-supervised inner task adapts the memory during an episode, while an outer objective learns the representations and update rule so that this adaptation improves downstream predictions. We first review these two learning processes and then discuss how temporal information can enter TTT.

The parameters adapted within an episode are called *fast weights*, denoted by $\mathbf{W}$, and parameterize a memory model $f_{\mathbf{W}} : \mathbb{R}^{d_k} \to \mathbb{R}^{d_v}$. The fast weights serve as the memory state and are updated as new observations arrive. An encoder and learned projections transform each observation o_t into a key $\mathbf{k}_t \in \mathbb{R}^{d_k}$ and a value $\mathbf{v}_t \in \mathbb{R}^{d_v}$. The key specifies the input to the memory model, while the value provides the supervision signal for the inner learning task.

A query $\mathbf{q}_t \in \mathbb{R}^{d_k}$, also computed from o_t, is used to read the memory for action prediction. The parameters optimized by the outer objective are called the *slow weights* θ. They include the encoder, projections, action network, and any learned components of the update rule. Unlike the fast weights, which adapt within each episode, the slow weights are shared across training sequences and updated through outer learning.

The two sets of parameters are learned through the following inner and outer processes. We write $\mathbf{W}_t$ and $\mathbf{W}_t^+$ for the fast weights before and after processing observation o_t.

1. **Inner learning: adapting the memory.** The observation available at step t defines a self-supervised learning objective ℓ_t^{inner}. The inner update changes only the fast weights:

$$\mathbf{W}_t^+ = \mathbf{W}_t - \eta \ \nabla_{\mathbf{W}} \ell_t^{\text{inner}}(\mathbf{W}; \theta)\big|_{\mathbf{W}=\mathbf{W}_t}, \tag{1}$$

   where $\eta > 0$ is the inner step size. The policy then combines the memory read $f_{\mathbf{W}_t^+}(\mathbf{q}_t)$ with observation features to predict an action, and the updated memory state is carried forward: $\mathbf{W}_{t+1} = \mathbf{W}_t^+$. This sequence of adaptation, reading, and carrying the updated weights allows TTT to maintain memory throughout an episode.

2. **Outer learning: learning through adaptation.** The outer objective trains the slow weights using action supervision after the memory has adapted:

$$\mathcal{L}^{\text{outer}}(\theta) = \mathbb{E}\left[\sum_{t=0}^{T-1} \ell_t^{\text{action}}\big(\theta, \mathbf{W}_t^+(\theta)\big)\right], \tag{2}$$

   where T denotes the length of the training sequence and the expectation is over training sequences and action-model randomness.
   The dependence $\mathbf{W}_t^+(\theta)$ arises because the representations and update rule controlled by θ determine how the memory is adapted. Differentiating the outer objective through the inner update therefore involves terms of the form

$$\frac{\partial}{\partial \theta}\left[\nabla_{\mathbf{W}} \ell_t^{\text{inner}}(\mathbf{W}; \theta)\right]\Big|_{\mathbf{W}=\mathbf{W}_t}. \tag{3}$$

   These derivatives allow outer supervision to optimize not only how the memory is read, but also what information is stored and how the memory changes in response to new observations. This corresponds to the gradient-based meta-learning view of TTT (Finn et al., 2017; Sun et al., 2025). During deployment, the slow weights are fixed, while the inner task continues adapting the memory from incoming observations.

Two temporal roles in TTT. Because the memory state evolves sequentially, temporal information can affect TTT in two distinct ways.

First, an action loss at time t can supervise a memory update made at an earlier time $u < t$ through the intervening fast-weight states. This is temporal credit assignment: later predictions provide supervision for earlier updates. Truncated backpropagation through time limits this optimization path to a gradient window of at most G updates (Aicher et al., 2020), while the numerical memory state continues across window boundaries. At $G = 1$, the current action loss still trains the current memory update; only gradients through earlier updates are removed.

Second, temporal information can affect the construction of the inner learning problem itself. The inner task at time t may use observations acquired at earlier times $u < t$ when defining the current adaptation objective. Unlike temporal credit assignment, this mechanism determines what information is used for memory adaptation rather than how far gradients propagate through the update chain.

3 Temporal Contrastive Test-Time Training

We introduce Temporal Contrastive Test-Time Training (TC–TTT), which extends test-time training with temporal inner supervision for Diffusion Policy. While TTT enables online adaptation through self-supervised memory updates, TC–TTT exploits temporal experience from sequential observations when constructing the inner learning signal. The following section first presents the overall framework and the integration of the adaptive memory module into Diffusion Policy. We then formulate the temporal contrastive objective, analyze the resulting update behavior, and discuss implementation details.

3.1 Overall Framework

Figure 2 shows how TC–TTT adds an adaptive memory to the standard RMBench Diffusion Policy backbone (Chen et al., 2026; Chi et al., 2025). Our controlled ablations share this backbone and memory interface, varying the inner objective, gradient window, or history sampling. Appendix B.1 specifies the inherited DP components and the added TTT module. At each timestep t, images and robot state are encoded into observation features, from which learned projections produce a key $\mathbf{k}_t$, a value $\mathbf{v}_t$, and a query $\mathbf{q}_t$. The Encoder block groups these encoding and projection operations. The memory model $f_{\mathbf{W}}$ stores experience in fast weights that evolve within the episode.

The inner task first updates this memory. Reconstruction learns the current key–value association, while temporal contrast uses an earlier key to distinguish the current value from up to K sampled past alternatives. Both objectives are evaluated at the same incoming fast weights and combined into one gradient step, producing $\mathbf{W}_t^+$. Removing temporal contrast recovers reconstruction-only TTT; the formal objective is defined in Section 3.2.

The query then reads the updated memory. This read is layer-normalized and concatenated with the current observation features, forming the condition for the diffusion U-Net that predicts an action chunk (Chi et al., 2025). The matching "obs" ports in the figure denote this direct feature connection. Memory is updated once per new observation; all diffusion denoising steps reuse the resulting condition. The updated fast weights become the next step's memory, and key/value features are cached for future temporal comparisons. Algorithm 1 gives the per-observation procedure.

During sequence training (Figure 2a), action supervision backpropagates through the memory read and inner update to train the slow weights. TBPTT limits this path to at most G memory updates, including the current one, while cached historical features are detached from the gradient graph. Thus $G = 1$ still trains the current update. The candidate count K controls inner supervision independently of this outer gradient window, and TBPTT can be used with either inner objective.

During inference (Figure 2b), the encoder, projections, and diffusion network remain fixed as part of the slow weights. Inner memory adaptation and history accumulation continue as new observations arrive, preserving the same update–read–act sequence used in training.

3.2 Temporal Contrastive Objective

TC–TTT constructs temporal supervision by relating previously observed inputs with information available at the current timestep. For each timestep t, we define the available historical indices as

$$\mathcal{H}_t = \{t - js : 1 \le j \le K,\ t - js \ge 0\}, \tag{4}$$

where s controls the temporal spacing and K limits the number of historical candidates. The key $\mathbf{k}_{t-s}$ from the most recent historical observation serves as the temporal input. The current value $\mathbf{v}_t$ serves as the positive target, while historical values $\{\mathbf{v}_i : i \in \mathcal{H}_t\}$ provide alternative candidates.

We define the inner objective of TC–TTT as

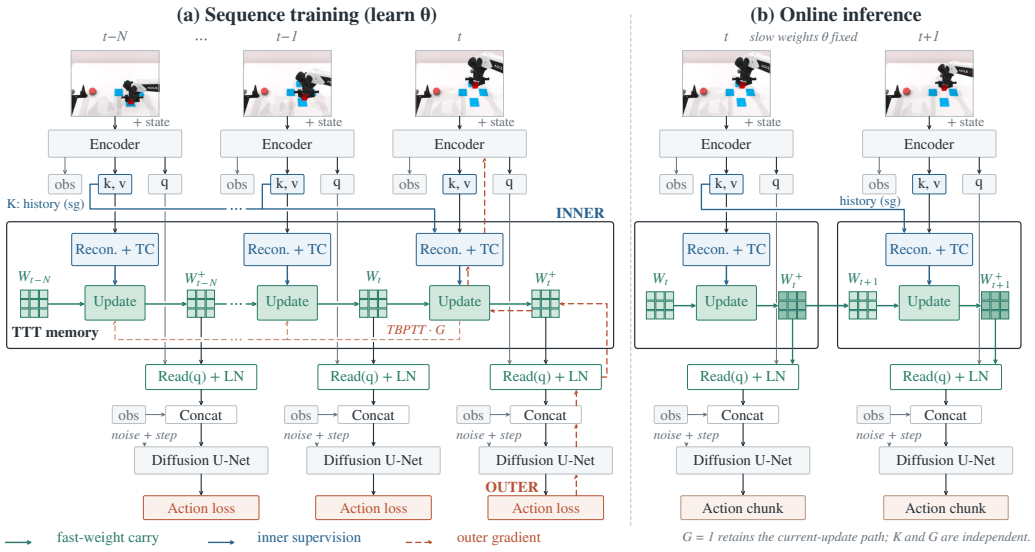


Figure 2: **TC–TTT training and inference. (a)** Action supervision trains slow weights through joint reconstruction and temporal contrast memory updates. **(b)** Inference fixes slow weights while memory adaptation continues. K: historical candidate count; G: outer gradient window; sg: stop-gradient; N: illustrative past offset. Images illustrate RMBench Put Back Block.

$$\ell_t^{\text{inner}}(\mathbf{W}) = \ell_t^{\text{rec}}(\mathbf{W}) + \lambda \ell_t^{\text{tc}}(\mathbf{W}), \tag{5}$$

where ℓ_t^{rec} denotes the reconstruction objective used in standard TTT and λ controls the contribution of temporal supervision. The temporal contrastive term uses an InfoNCE-style objective (van den Oord et al., 2018):

$$\ell_t^{\text{tc}}(\mathbf{W}) = -\log \frac{\exp(\text{sim}(f_{\mathbf{W}}(\mathbf{k}_{t-s}), \mathbf{v}_t)/\tau)}{\sum_{i \in \mathcal{H}_t \cup \{t\}} \exp(\text{sim}(f_{\mathbf{W}}(\mathbf{k}_{t-s}), \mathbf{v}_i)/\tau)}, \tag{6}$$

where $\text{sim}(\cdot, \cdot)$ denotes cosine similarity and τ is the temperature parameter. The temporal objective encourages the memory read from an earlier observation to retrieve the current target while distinguishing it from historical alternatives. When no historical observation is available, we set $\ell_t^{\text{tc}} = 0$.

The resulting inner objective is optimized using the TTT adaptation rule introduced in Section 2. This objective can be applied to a differentiable memory model $f_{\mathbf{W}}$; its definition does not require the particular matrix parameterization used in our experiments.

3.3 Theoretical Analysis

We analyze the local behavior of the TC–TTT update to understand how temporal contrastive supervision changes memory adaptation. Compared with reconstruction-only TTT, the temporal objective introduces an additional history-dependent correction term. The exact matrix update is derived in Appendix A.1. The following proposition characterizes this effect in a simplified setting: a matrix memory $f_{\mathbf{W}}(\mathbf{x}) = \mathbf{W}\mathbf{x}$, fixed keys and queries, and small perturbations about a constant value. The matrix parameterization matches the fast-weight component in our implementation; fixing the addresses and linearizing the dynamics are assumptions for analysis. The complete policy includes nonlinear encoding, read normalization, and diffusion action prediction.

Proposition 1 (History-dependent correction in temporal adaptation). *Let $f_{\mathbf{W}}(\mathbf{x}) = \mathbf{W}\mathbf{x}$, with all keys and queries fixed at a common unit address. For small perturbations about a constant value, let f_t and y_t be the first-order coefficients of the value and post-update memory read along a direction orthogonal to the reference value. Under the local assumptions in Appendix A.2, these coefficients satisfy*

$$y_t - y_{t-1} = \underbrace{\beta(f_t - y_{t-1})}_{\text{reconstruction correction}} + \underbrace{\kappa\left(f_t - \frac{1}{K}\sum_{j=1}^{K} f_{t-js}\right)}_{\text{history-dependent correction}}, \tag{7}$$

where $\beta = 2\eta/d_v$ and $\kappa = \eta\lambda K/[\tau(K+1)]$.

The first term corresponds to the standard reconstruction update, which reduces the prediction error between the current target and the memory readout. The second term compares the current representation with historical representations and therefore introduces an additional temporal signal into the memory update.

For smoothly varying representations, the history-dependent difference can be related to a temporal derivative. Letting $f_t = f(t)$, we obtain

$$f(t) - \frac{1}{K}\sum_{j=1}^{K} f(t - js) \approx \frac{s(K+1)}{2} f'(t). \tag{8}$$

Therefore, the temporal contrastive objective can be interpreted as introducing a velocity-like component in the learned representation space: the memory update is not only correcting current prediction errors, but also responding to temporal changes in the observation sequence. Appendix A.2 gives the full local assumptions and derivation underlying Proposition 1. Appendix A.3 gives the conditions and error bound for the derivative approximation in Equation 8.

4 Experiments

In this section, we empirically evaluate temporal comparison as a learning signal for online memory in robotic manipulation. Specifically, we address the following research questions:

- **Q1: Does temporal comparison provide an effective learning signal for online memory?**
- **Q2: How should history be sampled for temporal comparison?**
- **Q3: How does temporal comparison shape the memory update?**

Setup. We evaluate on RMBench (Chen et al., 2026). RMBench groups tasks by Task Memory Complexity (TMC): $M(1)$ tasks require retaining a single task-relevant past observation, whereas $M(n)$ tasks require integrating information from multiple task-relevant past observations.

We use task-specific Diffusion Policies (Chi et al., 2025), each with 77M parameters, built on the RMBench backbone with a TTT module (see Appendix B.1 for parameter accounting). We follow the default RMBench optimizer and diffusion settings; sequence-training details are provided in Appendix B. We evaluate each task with 100 rollouts and report per-task success rates and task-balanced mean success rates. Detailed results and evaluation protocols are provided in Appendix C.

4.1 Q1: Does temporal comparison provide an effective learning signal?

We assess effectiveness in two complementary ways. First, we compare TC–TTT with selected RMBench policies in the sub-billion-parameter regime, together with Mem-0 as a canonical memory-aware reference. Table 1 reports per-task results for selected RMBench policies; Appendix C.1 provides the broader benchmark comparison. Second, we examine the design choice central to TC–TTT: using temporal comparison to supervise the inner memory update. Under the same policy

Table 1: **Task-level success on RMBench (%).** TC–TTT uses $(s, K, G) = (16, 16, 1)$. Dashes denote unreported tasks. Means cover each method's available tasks; means over different task subsets are not directly comparable.

	$M(1)$						$M(n)$					
Policy	Obs.	Put	Rear.	S-Blk.	S-T	Avg.	Batt.	Rank	Cov.	Press	Avg.	Mean
DP (Chi et al., 2025)	1	0	0	11	20	6.4	10	10	0	0	5.0	5.8
TTT ($G = 16$)	2	88	**100**	24	75	57.8	38	36	3	0	19.3	40.7
MemoAct (Tan et al., 2026)	4	41	98	–	53	49.1	–	–	–	–	–	49.1
Mem-0 (Chen et al., 2026)	4	**90**	89	67	14	52.8	28	18	68	0	28.5	42.0
AR-K (Jiang et al., 2026a)	44	58	62	68	18	50.0	38	18	72	82	52.5	51.1
AR-Q (Jiang et al., 2026a)	**84**	88	86	**82**	16	**71.2**	**52**	34	**78**	**90**	**63.5**	**67.8**
TC–TTT (ours)	4	85	98	80	**81**	69.6	37	**55**	19	7	29.5	51.8

Note. Mem-0 adds a planner on $M(n)$ tasks; AR-K and AR-Q pair the same AR-WAM execution policy with different external agents. means use unrounded rates and retain one decimal place; See Appendix C.1 for evaluation details.

Table 2: **Inner history versus outer gradient reach.** Task-balanced success (%) for TTT across G and TC–TTT across K, with $G = s = 1$ for TC–TTT. Gain is the paired All-9 difference in percentage points. Dashes indicate an unpaired baseline.

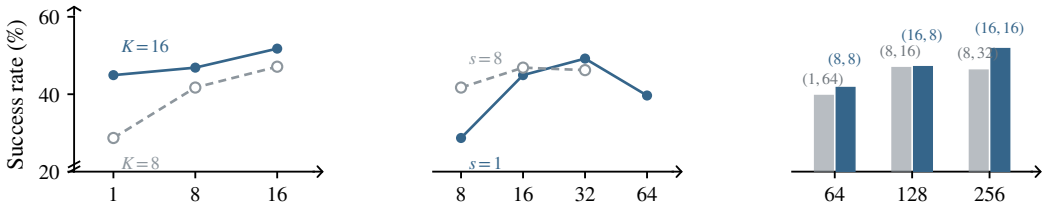
TTT: extend gradient window				**TC–TTT: extend history**				**Gain**
G	$M(1)$	$M(n)$	All 9	K	$M(1)$	$M(n)$	All 9	Δ All 9
1	41.4	19.6	31.8	—	—	—	—	—
16	57.8	19.3	40.7	16	63.0	22.3	44.9	+4.2
32	52.6	21.6	38.8	32	66.2	28.0	49.2	+10.4
64	53.4	18.0	37.7	64	50.0	26.8	39.7	+2.0

architecture, we compare extending the outer action-gradient window G in TTT with extending the inner comparison history K in TC–TTT.

Comparison with selected RMBench policies. TC–TTT reaches 51.8% mean success, exceeding reconstruction-only TTT trained with an outer gradient window of $G = 16$ by 11.1 percentage points (Table 1). The gains vary across tasks: Swap Blocks improves from 24% to 80%, while Put Back decreases from 88% to 85%. Published diffusion, memory-augmented, and world-action-model policies provide additional benchmark context; their architectures, task coverage, and evaluation protocols differ. Appendix C.1 provides the full comparison details.

Inner comparison versus outer action gradients. Table 2 contrasts two temporal axes of TTT optimization. Increasing G extends the outer credit-assignment horizon, allowing the action loss to backpropagate through more past memory updates. Increasing K instead extends the history available to the inner temporal-comparison objective, while keeping the action-gradient window at $G = 1$ and the history stride at $s = 1$. We compare the two extensions at the same numerical settings (16, 32, and 64). Here G counts updates including the current one, whereas K counts past candidates; these settings therefore do not specify identical temporal extents.

Extending the outer gradient window alone does not consistently improve success: reconstruction-only TTT peaks at $G = 16$ among the tested settings. TC–TTT exceeds the numerically paired TTT settings at all three tested values while retaining $G = 1$ (Table 2), with a maximum gain of 10.4 percentage points in mean success. Appendix C.2 gives the corresponding per-task scores. Together, these results support the central design of TC–TTT: historical observations provide an effective supervisory signal when used directly in the inner memory objective, even while the action-gradient window remains short.



(a) Stride ($K = 8, 16$). **(b)** Candidates ($s = 1, 8$). **(c)** Equal maximum age.

Figure 3: **History sampling at $G = 1$.** Nine-task mean success (%). Panels vary **(a)** stride, **(b)** candidate count, and **(c)** sampling density at matched maximum age sK; bar labels give (s, K). Appendix C.2 reports the corresponding task scores.

4.2 Q2: How should history be sampled for temporal comparison?

We next study how to construct the historical candidate set used by temporal comparison. Its sampling is controlled by two axes: the stride s, which determines the temporal spacing between historical observations, and the candidate count K, which determines the maximum number of observations entering the inner comparison objective. All configurations use $G = 1$.

Increasing stride improves the displayed mean in both fixed-K scans, $K = 8$ and $K = 16$ (Figure 3a), whose training and evaluation settings match within each scan. Increasing the candidate count is not uniformly beneficial: the means peak at an intermediate K in each scan and then fall (Figure 3b).

Figure 3c compares sampling densities at matched maximum history age sK. Sparser sampling scores higher at ages 64 and 256, while the age-128 pair is nearly tied. Moreover, changing s also changes the historical input $\mathbf{k}_{t-s}$ and the number of available candidates early in an episode.

Together, these results favor exploring histories that cover a broad temporal range with separated observations. One possible explanation for the candidate-count trends is an increased risk of false negatives: larger or denser candidate sets may include observations that are semantically equivalent to the current state but treated as negatives (Chuang et al., 2020). Table 7 in Appendix C.2 gives the per-task scores for all plotted configurations.

4.3 Q3: How does temporal comparison shape the memory update?

Interventions on memory updates. We test the local derivative interpretation (Equation 8) by replacing the current TC write of frozen $G = 1, K = 16$ policies. Each intervention starts from native memory and evaluates the original diffusion head on held-out expert trajectories. Table 3 compares slope, secant, and norm-matched replacements against write removal.

Table 3: Current-write replacements. Task-balanced mean change in normalized expert-action MSE (%) relative to native TC (0), shown to two decimal places; lower is better.

TC replacement	$s = 1$	$s = 8$	$s = 16$
Removed	+29.85%	+115.79%	+124.58%
Local slope (9f)	+2.60%	+378.63%	+365.37%
Full-window slope	+6.85%	+181.17%	+57.35%
Local dir., window norm	+5.53%	+80.74%	+57.17%
Endpoint secant	+6.29%	+90.82%	+38.81%

Dependence on temporal span. At $s = 1$, the local slope increases mean action MSE by 2.60%, compared with 29.85% when the current TC write is removed (Table 3), supporting the short-span interpretation. At $s = 8, 16$, the local replacement deteriorates. Full-window and secant estimates reduce this error, although full-window OLS remains worse than removal at $s = 8$. Matching the local direction to the full-window write norm also substantially reduces error at both larger spans. The improvement therefore cannot be attributed to direction alone: accumulated change magnitude is an important part of the longer-span functional approximation. Appendix C.2 reports all nine task-level errors for these replacements and specifies their evaluation procedure.

5 Related Work

Memory-Augmented Robot Policies. Robot policies can use history by retaining selected observations, compressing past experience, or maintaining a recurrent memory state. MEM combines recent video with longer-term textual memory (Torne et al., 2026), BPP selects keyframes corresponding to meaningful events (Mark et al., 2026), and MemoAct separates lossless short-term storage from compressed long-term memory (Tan et al., 2026). AEM instead pretrains temporal representations by reconstructing masked vision–action histories (Zhou et al., 2026). More recent approaches learn compact states that scale to longer histories: SeedPolicy uses a gated recurrent state to compress long observation horizons (Gui et al., 2026), while related state-space and retrieval-based policies summarize or selectively access past experience for long-horizon control. These methods primarily address how history is represented, compressed, or retrieved. TC–TTT instead focuses on how relations within the observed history can supervise the update of an online parametric memory.

Test-Time Training. The idea of storing temporary information in adaptive parameters dates back to fast-weight models (Schmidhuber, 1992), with later associative-memory formulations using error-correcting key–value updates such as the delta rule (Schlag et al., 2021). Test-time training (TTT) introduced self-supervised parameter updates at inference time for adaptation under distribution shift (Sun et al., 2020). TTT layers subsequently turned this principle into a sequence model whose recurrent state is itself a learned model, updated online as new tokens arrive (Sun et al., 2025). This view connects TTT to a broader family of online parametric memories, including delta-rule sequence models (Yang et al., 2024; 2025; Siems et al., 2025), regression-based associative memory (Wang et al., 2025), and long-term neural memory mechanisms such as Titans (Behrouz et al., 2025b). Recent TTT work has increasingly focused on how the inner update should be parameterized and supervised. TTT-E2E uses next-token prediction to learn directly from the observed context (Tandon et al., 2025), In-Place TTT adapts parameters already present in a Transformer with an objective aligned to autoregressive prediction (Feng et al., 2026a), and contrastive TTT explores discriminative inner objectives (Anonymous, 2025; Liu et al., 2021). In robotics, WAM-TTT updates an adaptive memory inside a frozen world-action model from human video through self-supervised prediction (Feng et al., 2026b), while RoboTTT integrates TTT layers into robot policies so that long visuomotor histories are compressed into fast weights (Jiang et al., 2026b). Outside test-time training, time-contrastive learning has long been used to obtain robot representations from video by contrasting frames at different times (Sermanet et al., 2018). We place that comparison inside the memory update itself rather than in a representation-learning stage. TC–TTT builds on this parametric-memory view but changes the temporal supervision of the inner update: an earlier key is associated with the current value, while sampled historical values provide alternatives. The update uses only already observed information and is trained jointly with the outer action objective. We study how this temporal supervision interacts with history selection within a fixed robot policy architecture.

6 Conclusion

TC–TTT supervises online memory with historical comparisons. On nine RMBench tasks, our method raises mean success from 31.8% to 51.8% with the same one-update gradient window, demonstrating the value of historical inner supervision. Theoretical analysis (Section 3.3) and the Q3 interventions (Section 4.3) support a velocity-like interpretation at $s = 1$, but the approximation deteriorates at larger strides. Full-window and norm-matching results suggest a role for accumulated representational change, offering a direction for understanding the gains. A fuller explanation and validation beyond task-specific Diffusion Policies on nine simulated RMBench tasks, including larger models and larger-scale training, remain for future work.

AI USE STATEMENT

We used generative AI tools to assist with drafting and restructuring the manuscript, editing its language, refining mathematical arguments and exposition, discussing the method and interpreting results, checking literature sources and references, iterative manuscript review, and writing visualization scripts. The authors take responsibility for the final text, claims, analyses, and artifacts.

ETHICS STATEMENT

This work studies adaptive memory in simulated robotic manipulation tasks. Physical deployment would require task-specific safety controls and assessment of interactions with people and objects.

REPRODUCIBILITY STATEMENT

Section 3 specifies the memory objective, historical sampling, update, and training procedure. Appendix A states the assumptions and provides the derivations. Appendix B reports the architecture, initialization, demonstration split, training settings, checkpoint selection, and evaluation protocol. Appendix C provides additional results.

REFERENCES

Christopher Aicher, Nicholas J. Foti, and Emily B. Fox. Adaptively truncating backpropagation through time to control gradient bias. In *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pp. 799–808, 2020. URL https://proceedings.mlr.press/v115/aicher20a.html.

Anonymous. Contrastive learning for test-time training layers, 2025. URL https://openreview.net/forum?id=wZYi4faSGK. Anonymous manuscript submitted to TMLR.

H. B. Barlow. Possible principles underlying the transformations of sensory messages. In W. A. Rosenblith (ed.), *Sensory Communication*, pp. 217–234. MIT Press, 1961. doi: 10.7551/mitpress/9780262518420.003.0013.

Ali Behrouz, Zeman Li, Praneeth Kacham, Majid Daliri, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. Atlas: Learning to optimally memorize the context at test time. *arXiv preprint arXiv:2505.23735*, 2025a.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *Advances in Neural Information Processing Systems*, volume 38, 2025b. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a4ca07aa108036f80cbb5b82285fd4b1-Abstract-Conference.html.

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/2603.01229v3.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10–11):1684–1704, 2025. doi: 10.1177/02783649241273668. URL https://doi.org/10.1177/02783649241273668.

Ching-Yao Chuang, Joshua W. Robinson, Yen-Chen Lin, Antonio Torralba, and Stefanie Jegelka. Debiased contrastive learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/63c3ddcc7b23daa1e42dc41f9a44a873-Abstract.html.

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.

Guhao Feng, Shengjie Luo, Kai Hua, Ge Zhang, Wenhao Huang, Di He, and Tianle Cai. In-place test-time training. In *International Conference on Learning Representations*, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/ba6f5879d7f382241316f7dda7e6b8ac-Abstract-Conference.html.

Yusen Feng, Bingchen Han, Jiangran Lyu, Kai Liu, Yixin Zheng, Yuxuan Wan, Weiheng Liu, Sun Han, Ruiqin Li, Yulong Zhang, Fangfu Liu, Xuesong Shi, Libin Liu, Yizhou Wang, Zhizheng Zhang, and He Wang. WAM-TTT: Steering world-action models by watching human play at test time. *arXiv preprint arXiv:2607.06988*, 2026b. URL https://arxiv.org/abs/2607.06988.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135, 2017. URL https://proceedings.mlr.press/v70/finn17a.html.

Youqiang Gui, Yuxuan Zhou, Shen Cheng, Xinyang Yuan, Haoqiang Fan, Peng Cheng, and Shuaicheng Liu. SeedPolicy: Horizon scaling via self-evolving diffusion policy for robot manipulation. *arXiv preprint arXiv:2603.05117*, 2026. URL https://arxiv.org/abs/2603.05117.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016. URL https://arxiv.org/abs/1512.03385.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020. URL https://arxiv.org/abs/2006.11239.

Yicheng Jiang, Zesen Gan, Xiaobo Wang, Tianlun He, Chenxu Zhao, Minghui Wu, Xinyue Wang, Jiaxu Wang, Junhao He, Jianan Wang, and Qiming Shao. AR-WAM: A visual-conditioned agent-ready world action model for robotic manipulation. *arXiv preprint arXiv:2609.23578*, 2026a. URL https://arxiv.org/abs/2609.23578.

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

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998. doi: 10.1016/S0004-3702(98)00023-X.

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.

Peiyan Li, Yuze Zhu, Yixiang Chen, Qisen Ma, Yuan Xu, Jiabing Yang, He Guan, Yan Huang, Hongtao Wu, Xiao Ma, Tao Kong, Liang Wang, and Tieniu Tan. BridgeVLA++: A data-efficient, generalizable, and memory-augmented vision-language-action framework for 3D manipulation. *arXiv preprint arXiv:2608.05042*, 2026. URL https://arxiv.org/abs/2608.05042v1.

Yuejiang Liu, Parth Kothari, Bastien 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 (NeurIPS)*, 2021.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1711.05101.

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.

Rajesh P. N. Rao and Dana H. Ballard. Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2(1):79–87, 1999. doi: 10.1038/4580. URL https://doi.org/10.1038/4580.

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

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992. URL https://people.idsia.ch/~juergen/fastweights/.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. In *IEEE International Conference on Robotics and Automation*, 2018. URL https://arxiv.org/abs/1704.06888.

Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. DeltaProduct: Improving state-tracking in linear RNNs via householder products. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2502.10297.

M. V. Srinivasan, S. B. Laughlin, and A. Dubs. Predictive coding: A fresh view of inhibition in the retina. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 216(1205): 427–459, 1982. doi: 10.1098/rspb.1982.0085. URL https://pubmed.ncbi.nlm.nih.gov/6129637/.

Xiaoquan Sun, Ruijian Zhang, Chen Cao, Yihan Sun, Jiahui Chen, Zetian Xu, Bo Chen, Haijier Chen, Zhen Yang, Jiarun Zhu, Yijun Hong, JingZhe Xu, Jingrui Pang, Mingqi Yuan, and Jiayu Chen. HiMem-WAM: Hierarchical memory-gated world action models for robotic manipulation. *arXiv preprint arXiv:2606.10363*, 2026.

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, 2020. URL https://proceedings.mlr.press/v119/sun20b.html.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): RNNs with expressive hidden states. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 57503–57522, 2025. URL https://proceedings.mlr.press/v267/sun25h.html.

Liufan Tan, Jiale Li, and Gangshan Jing. MemoAct: Atkinson–shiffrin-inspired hierarchical memory-augmented policy for robotic manipulation. *IEEE Robotics and Automation Letters*, 2026. doi: 10.1109/LRA.2026.3719233.

Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, Carlos Guestrin, Jed McCaleb, Yejin Choi, and Yu Sun. End-to-end test-time training for long context. *arXiv preprint arXiv:2512.23675*, 2025. URL https://arxiv.org/abs/2512.23675.

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.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. URL https://arxiv.org/abs/1807.03748.

Johannes von Oswald, Nino Scherrer, Seijin Kobayashi, Luca Versari, Songlin Yang, Sarthak Mittal, Maximilian Schlegel, Kaitlin Maile, Yanick Schimpf, Oliver Sieberling, Alexander Meulemans, Rif A. Saurous, Guillaume Lajoie, Charlotte Frenkel, Razvan Pascanu, Blaise Agüera y Arcas, and João Sacramento. MesaNet: Sequence modeling by locally optimal test-time training. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2506.05233.

Kai Wang, Zhaopeng Gu, Yixiang Chen, Yuan Xu, Qisen Ma, Peng Su, Zhaowen Li, Yan Huang, and Liang Wang. DiM-WAM: World-action modeling with diverse historical event memory. *arXiv preprint arXiv:2606.27677*, 2026.

Ke Alexander Wang, Jiaxin Shi, and Emily B. Fox. Test-time regression: A unifying framework for designing sequence models with associative memory. *arXiv preprint arXiv:2501.12352*, 2025. URL https://arxiv.org/abs/2501.12352.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision*, pp. 3–19, 2018. URL https://arxiv.org/abs/1803.08494.

Fan Yang, Yuting Su, Xiaobo Wang, Yuncheng You, Fugui Fan, Yuting Wu, Minghui Wu, Chenxu Zhao, JiaHong Ning, and Peiguang Jing. LiLa-WAM: Lightweight latent reasoning world-action model for robotic manipulation. *arXiv preprint arXiv:2608.03701*, 2026a. URL https://arxiv.org/abs/2608.03701v1.

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*, 2026b. URL https://arxiv.org/abs/2606.20092v2.

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*, 2026c. URL https://arxiv.org/abs/2606.20562v1.

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

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2412.06464.

Cheng Yin, Wang Xu, Junpeng Yang, Sikyuen Tam, Hanyu Liu, Yuan Yao, Xiangrui Zeng, Junbo Cui, Yequan Wang, Zhouping Yin, and Yankai Lin. SimpleMemVLA: A simple but effective native-video memory for vision-language-action models. *arXiv preprint arXiv:2609.05533*, 2026. URL https://arxiv.org/abs/2609.05533v1.

Youngju Yoo, Jiaheng Hu, Yifeng Zhu, Bo Liu, Qiang Liu, Roberto Martín-Martín, and Peter Stone. RoboSSM: Scalable in-context imitation learning via state-space models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2026. URL https://arxiv.org/abs/2509.19658.

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*, 2026. URL https://arxiv.org/abs/2603.16666v1.

Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T. Freeman, and Hao Tan. Test-time training done right. *arXiv preprint arXiv:2505.23884*, 2025. URL https://arxiv.org/abs/2505.23884. Large-chunk test-time training (LaCT).

Sizhe Zhao, Haozhe Xie, Weiyu Zhao, Chenchu Zhang, Huan Wang, Chenyang Wang, Qinglin Liu, and Shengping Zhang. Memory as plans: World-action modeling with memory-grounded planning. *arXiv preprint arXiv:2609.11561*, 2026.

Yijing Zhou, Qiwei Liang, Sitong Zhuang, Jiaxi Li, Xianpeng Wang, Boyang Cai, Yunyang Mo, and Renjing Xu. Action-effect memory pretraining for robot manipulation. *arXiv preprint arXiv:2606.12499*, 2026. URL https://arxiv.org/abs/2606.12499.

A Derivations of the Memory Update

This appendix develops the update analysis in Section 3.3 in three steps. Appendix A.1 derives the exact gradient of the joint objective for $f_{\mathbf{W}}(\mathbf{x}) = \mathbf{W}\mathbf{x}$, the inner predictor used in our TTT module. Appendix A.2 specifies the local setting behind Proposition 1, derives its recurrence, and controls the approximation error over time. Appendix A.3 then relates the historical difference to a temporal derivative and bounds the error of that approximation. The fixed-address and small-perturbation assumptions belong to the local analysis; they do not describe the full nonlinear policy in Appendix B.1.

A.1 Exact Gradient of the Joint Objective

We derive the matrix update induced by the objective in Section 3.2. Throughout this subsection, $f_{\mathbf{W}}(\mathbf{x}) = \mathbf{W}\mathbf{x}$, and the encoded keys and values are held fixed when differentiating with respect to $\mathbf{W}$.

Normalization and candidate representations. We use $n_\epsilon(\mathbf{z}) = \mathbf{z}/\max(\|\mathbf{z}\|_2, \epsilon)$, with $\epsilon > 0$, to normalize keys, queries, and the arguments of cosine similarity. The similarity is $\mathrm{sim}(\mathbf{x}, \mathbf{v}) = n_\epsilon(\mathbf{x})^\top n_\epsilon(\mathbf{v})$. For the historical key $\mathbf{h}_t = \mathrm{sg}(\mathbf{k}_{t-s})$, write $\mathbf{x}_t = \mathbf{W}_t\mathbf{h}_t$, $\mathbf{u}_t = n_\epsilon(\mathbf{x}_t)$, and $\mathbf{c}_i = n_\epsilon(\mathbf{v}_i)$. The Jacobian of normalization, away from $\|\mathbf{x}\|_2 = \epsilon$, is

$$J_\epsilon(\mathbf{x}) = \begin{cases} (\mathbf{I} - \mathbf{u}\mathbf{u}^\top)/\|\mathbf{x}\|_2, & \|\mathbf{x}\|_2 > \epsilon, \\ \mathbf{I}/\epsilon, & \|\mathbf{x}\|_2 < \epsilon, \end{cases} \qquad \mathbf{u} = n_\epsilon(\mathbf{x}). \tag{9}$$

Temporal comparison gradient. For $m = |\mathcal{H}_t| \geq 1$, define probabilities over the current and historical candidates by

$$p_i = \frac{\exp(\mathbf{u}_t^\top \mathbf{c}_i/\tau)}{\sum_{j \in \mathcal{H}_t \cup \{t\}} \exp(\mathbf{u}_t^\top \mathbf{c}_j/\tau)}, \qquad p_{+,t} = p_t. \tag{10}$$

Normalizing over the historical candidates alone gives $\alpha_{t,i} = p_i/(1 - p_{+,t})$ for $i \in \mathcal{H}_t$. Their weighted average and the present–history difference are

$$\bar{\mathbf{c}}_t = \sum_{i \in \mathcal{H}_t} \alpha_{t,i}\mathbf{c}_i, \qquad \mathbf{e}_t = \mathbf{c}_t - \bar{\mathbf{c}}_t. \tag{11}$$

Differentiating the log-softmax loss gives

$$\frac{\partial \ell_{\mathrm{tc},t}}{\partial \mathbf{u}_t} = \frac{1}{\tau}\left(\sum_i p_i\mathbf{c}_i - \mathbf{c}_t\right) = -\frac{1 - p_{+,t}}{\tau}\mathbf{e}_t. \tag{12}$$

Applying the chain rule through the normalized memory read yields the temporal contribution

$$\Delta\mathbf{W}_t^{\mathrm{tc}} = \frac{\eta\lambda(1 - p_{+,t})}{\tau} J_\epsilon(\mathbf{x}_t)^\top \mathbf{e}_t\mathbf{h}_t^\top. \tag{13}$$

Joint update and query response. Using the mean squared reconstruction loss $\ell_{\mathrm{rec},t} = \|\mathbf{W}\mathbf{k}_t - \mathbf{v}_t\|_2^2/d_v$ and writing $\beta = 2\eta/d_v$, the exact joint update is

$$\mathbf{W}_t^+ - \mathbf{W}_t = \beta(\mathbf{v}_t - \mathbf{W}_t\mathbf{k}_t)\mathbf{k}_t^\top + \frac{\eta\lambda(1 - p_{+,t})}{\tau} J_\epsilon(\mathbf{x}_t)^\top \mathbf{e}_t\mathbf{h}_t^\top. \tag{14}$$

The two terms use different addresses: reconstruction uses the current key, and the temporal comparison uses the selected historical key. The latter contributes to a query $\mathbf{q}$ as

$$\Delta\mathbf{W}_t^{\mathrm{tc}}\mathbf{q} = \frac{\eta\lambda(1 - p_{+,t})}{\tau} J_\epsilon(\mathbf{x}_t)^\top \mathbf{e}_t(\mathbf{h}_t^\top \mathbf{q}). \tag{15}$$

The key overlap determines how strongly this update affects the query. Above the normalization floor, the Jacobian projects the present–history difference onto the subspace orthogonal to the incoming read and rescales it by the read's inverse norm.

A.2 Local Dynamics of the Memory Read

We make the fixed-address, small-perturbation setting of Proposition 1 explicit. The first part of the derivation establishes the recurrence in Equation 7. The second shows that its approximation to the full memory read remains accurate over time.

Local setting. Let all keys, historical addresses, and queries equal a unit vector $\mathbf{k}$, and let $\mathbf{v}_t = \mathbf{c}_* + \delta f_t \mathbf{d}_*$, where $\mathbf{c}_*$ and $\mathbf{d}_*$ are orthogonal unit vectors and f_t is uniformly bounded, including the prehistory. Assume a full K-candidate history at ages $s, \dots, Ks$, fixed $\eta > 0$, $\lambda \geq 0$, $\tau > 0$, integers $s, K \geq 1$, $0 < \beta = 2\eta/d_v < 1$, and $0 < \epsilon < 1$. Initialize the raw memory read as $\mathbf{z}_{-1} = \mathbf{c}_* + \delta y_{-1}\mathbf{d}_* + O(\delta^2)$ with bounded initialization constants.

Approximation statement. For sufficiently small $|\delta|$, the post-update read $\mathbf{z}_t = \mathbf{W}_t^+ \mathbf{k}$ satisfies $\mathbf{z}_t = \mathbf{c}_* + \delta y_t \mathbf{d}_* + O(\delta^2)$ uniformly in time, where y_t follows Equation 7 with $\kappa = \eta\lambda K/[\tau(K+1)]$.

Derivation and error control. Step 1: the first-order recurrence. Write $r = 1-\beta$. Since all addresses coincide, the incoming read at step t is the preceding post-update read $\mathbf{z}_{t-1}$. Equation 14 gives

$$\mathbf{z}_t = r\mathbf{z}_{t-1} + \beta \mathbf{v}_t + \frac{\eta\lambda(1-p_{+,t})}{\tau} J_\epsilon(\mathbf{z}_{t-1})^\top \left(\mathbf{c}_t - \sum_{j=1}^{K} \alpha_{t,t-js}\mathbf{c}_{t-js} \right). \tag{16}$$

At $\delta = 0$, all candidates equal $\mathbf{c}_*$, the temporal gradient is zero, and $\mathbf{z} = \mathbf{c}_*$ is a fixed point. For a read $\mathbf{z} = \mathbf{c}_* + \delta y \mathbf{d}_* + O(\delta^2)$, normalization and orthogonality imply

$$\begin{aligned} \mathbf{c}_i &= \mathbf{c}_* + \delta f_i \mathbf{d}_* + O(\delta^2), & n_\epsilon(\mathbf{z}) &= \mathbf{c}_* + \delta y \mathbf{d}_* + O(\delta^2), \\ n_\epsilon(\mathbf{z})^\top \mathbf{c}_i &= 1 + O(\delta^2), & \alpha_{t,t-js} &= \frac{1}{K} + O(\delta^2), \\ 1 - p_{+,t} &= \frac{K}{K+1} + O(\delta^2), & J_\epsilon(\mathbf{z})^\top \mathbf{d}_* &= \mathbf{d}_* + O(\delta). \end{aligned} \tag{17}$$

The temporal term in Equation 16 is therefore

$$\kappa\delta \left(f_t - \frac{1}{K}\sum_{j=1}^{K} f_{t-js} \right) \mathbf{d}_* + O(\delta^2). \tag{18}$$

Equating first-order coefficients gives Equation 7.

Step 2: a uniform approximation error. To establish a uniform remainder, fix constants M, Y, A with $\sup_{t \geq -Ks} |f_t| \leq M$, $|y_{-1}| \leq Y$, and $\|\mathbf{z}_{-1} - \mathbf{c}_* - \delta y_{-1}\mathbf{d}_*\|_2 \leq A\delta^2$. Denote the exact map in Equation 16 by

$$F(\mathbf{z}, \delta, \mathbf{a}) = r\mathbf{z} + \beta(\mathbf{c}_* + \delta a_0 \mathbf{d}_*) + B(\mathbf{z}, \delta, \mathbf{a}),$$

where $\mathbf{a} = (f_t, f_{t-s}, \dots, f_{t-Ks})$ and B is the temporal term. Choose a closed ball of radius $0 < \rho < 1 - \epsilon$ about $\mathbf{c}_*$. Normalization is smooth there, and $B(\mathbf{z}, 0, \mathbf{a}) = 0$ throughout the ball. Compactness gives constants C_0, C_1 such that

$$\|B\|_2 \leq C_0|\delta|, \qquad \|D_{\mathbf{z}} B\|_{\mathrm{op}} \leq C_1|\delta| \tag{19}$$

uniformly over $\mathbf{a} \in [-M, M]^{K+1}$ for small $|\delta|$. Choose $L > \max\{Y, (\beta M + C_0)/\beta\}$. For sufficiently small amplitude, the initial read lies within distance $L|\delta|$ of $\mathbf{c}_*$ and

$$\|F(\mathbf{z}, \delta, \mathbf{a}) - \mathbf{c}_*\|_2 \leq r\|\mathbf{z} - \mathbf{c}_*\|_2 + (\beta M + C_0)|\delta| \leq L|\delta|. \tag{20}$$

Hence every subsequent read stays in this neighborhood. Reducing the amplitude if necessary makes the exact map Lipschitz with constant $q \leq 1 - \beta/2$ on the fixed ball.

The scalar recurrence gives $|y_t| \leq Y + (\beta + 2\kappa)M/\beta$. Its corresponding path $\bar{\mathbf{z}}_t = \mathbf{c}_* + \delta y_t \mathbf{d}_*$ therefore stays in the same fixed ball for small amplitude. Taylor expansion yields a uniform one-step error $\|F(\bar{\mathbf{z}}_{t-1}, \delta, \mathbf{a}_t) - \bar{\mathbf{z}}_t\|_2 \leq D\delta^2$. For $E_t = \|\mathbf{z}_t - \bar{\mathbf{z}}_t\|_2$, we obtain $E_t \leq qE_{t-1} + D\delta^2$, and thus

$$\sup_{t \geq 0} E_t \leq \left(A + \frac{2D}{\beta} \right) \delta^2. \tag{21}$$

□

A.3 Derivative Approximation and Its Error

This final step justifies the derivative approximation in Equation 8. Starting from the historical difference in the local recurrence, we use Taylor's theorem to identify its first-order term and the curvature-dependent remainder.

Let f be twice continuously differentiable on $[t - Ks, t]$. Taylor's theorem gives $f(t - js) = f(t) - jsf'(t) + \frac{1}{2}(js)^2 f''(\xi_j)$ for some $\xi_j \in (t - js, t)$. Averaging over the history yields

$$f(t) - \frac{1}{K}\sum_{j=1}^{K} f(t - js) = \frac{s(K+1)}{2} f'(t) + R_t,$$
$$|R_t| \le \frac{s^2(K+1)(2K+1)}{12} \sup_{u \in [t-Ks,t]} |f''(u)|. \tag{22}$$

Equation 8 retains the first-order term when the value variation has small curvature over the sampled history. This is the time derivative of variation in the learned value representation.

B Policy Implementation and Experimental Protocols

This appendix separates the implementation of the policy from the protocols used to train and evaluate it. Appendices B.1 and B.2 describe the architecture and online memory state. Appendix B.3 summarizes the training recipe and its sequence-training settings. Appendix B.4 describes the shared 100-rollout evaluation protocol.

B.1 Policy Architecture

Our policy builds on the standard RMBench Diffusion Policy implementation (Chen et al., 2026; Chi et al., 2025). All controlled ablations in this paper share its visual backbone, diffusion action head, and the TTT interface described below. The ablations vary the memory objective, outer gradient window, or history construction.

RMBench DP backbone. Each observation contains a 96×96 RGB image from the head camera and a 14-dimensional vector of joint and gripper states. We retain the RMBench ResNet-18 encoder (He et al., 2016) with GroupNorm (Wu & He, 2018) and its conditional one-dimensional diffusion U-Net. The U-Net uses channel widths $(256, 512, 1024)$, kernel size 5, eight normalization groups, and a 128-dimensional diffusion-step embedding. It predicts noise for eight 14-dimensional actions, using the RMBench DDPM configuration with 100 diffusion steps and a squared-cosine noise schedule.

Observation conditioning. To connect this backbone to TTT, we concatenate the encoder's 512-dimensional image feature with the robot state and apply a linear layer, LayerNorm, and a Mish activation to obtain a 128-dimensional observation representation. The action condition combines this current representation with a memory read accumulated over the episode.

TTT module. Three learned projections without biases map the observation representation to 64-dimensional keys, values, and queries. Keys and queries use $n_\epsilon(\mathbf{x}) = \mathbf{x}/\max(\|\mathbf{x}\|_2, \epsilon)$; values remain unnormalized for reconstruction and are normalized when computing contrastive scores. The episode-specific fast weights form a 64×64 matrix: the inner predictor is $f_{\mathbf{W}}(\mathbf{x}) = \mathbf{W}\mathbf{x}$. After the inner update, the policy reads $\mathrm{LayerNorm}(\mathbf{W}_t^+ \mathbf{q}_t)$. Thus the fast predictor is a linear map, while the complete TTT policy includes nonlinear encoding, read normalization, and action prediction. The slow weights include the projections, read normalization, and a learned positive inner step size η.

Memory-conditioned action prediction. This 64-dimensional read and the observation representation form a 192-dimensional global condition for the diffusion U-Net. At inference, the condition stays fixed throughout the 100 denoising steps that produce the action sequence.

Algorithm 1 TC–TTT: one online observation step

Require: Observation o_t, fast weights $\mathbf{W}_t$, detached history cache $\mathcal{C}_t$, stride s, candidate limit K, slow weights θ
Ensure: Action chunk $\mathbf{A}_t$, next fast weights $\mathbf{W}_{t+1}$, next cache $\mathcal{C}_{t+1}$
if a new episode starts **then**
  $\mathbf{W}_t \leftarrow \mathbf{0}, \mathcal{C}_t \leftarrow \varnothing$
end if
$\mathbf{z}_t \leftarrow \mathrm{Encode}_\theta(o_t)$
$\mathbf{k}_t \leftarrow n_\epsilon(P_k\mathbf{z}_t)$, $\mathbf{v}_t \leftarrow P_v\mathbf{z}_t$, $\mathbf{q}_t \leftarrow n_\epsilon(P_q\mathbf{z}_t)$
$\mathcal{H}_t \leftarrow \{t - js : 1 \le j \le K,\ t - js \ge 0\}$
$\ell^{\mathrm{rec}}(\mathbf{W}) \leftarrow \|f_{\mathbf{W}}(\mathbf{k}_t) - \mathbf{v}_t\|_2^2/d_v$
$\ell^{\mathrm{tc}}(\mathbf{W}) \leftarrow 0$
if $\mathcal{H}_t \neq \varnothing$ **then**
  Retrieve $\mathbf{k}_{t-s}$ and $\{\mathbf{v}_i : i \in \mathcal{H}_t\}$ from $\mathcal{C}_t$
  Form $\ell^{\mathrm{tc}}(\mathbf{W})$ using Equation 6 with current target $\mathbf{v}_t$
end if
$\mathbf{W}_t^+ \leftarrow \mathbf{W}_t - \eta\ \nabla_{\mathbf{W}}[\ell^{\mathrm{rec}}(\mathbf{W}) + \lambda\ell^{\mathrm{tc}}(\mathbf{W})]\big|_{\mathbf{W}=\mathbf{W}_t}$
$\mathbf{m}_t \leftarrow \mathrm{LayerNorm}(f_{\mathbf{W}_t^+}(\mathbf{q}_t))$, $\mathbf{c}_t \leftarrow [\mathbf{z}_t; \mathbf{m}_t]$
$\mathbf{A}_t \leftarrow \mathrm{DDPMSample}_\theta(\mathbf{c}_t)$ ▷ fixed condition for all 100 steps
Append $\mathrm{sg}(\mathbf{k}_t), \mathrm{sg}(\mathbf{v}_t)$ to the cache; retain at most s keys and sK values as $\mathcal{C}_{t+1}$
$\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t^+$

Parameter accounting. Throughout this paper, "77M" denotes the 77,023,951 learned slow parameters in a single task-specific policy, rounded in decimal millions. We obtain this count by constructing the policy from the archived training configuration and summing the number of elements in its registered model parameters. All counted parameters are trainable during policy training.

Table 4: **Parameter count for one policy.** Reconstruction-only TTT and TC–TTT share this architecture. Episode-specific fast weights and history caches are excluded.

Component	Learned parameters
ResNet-18 visual encoder with GroupNorm	11,176,512
Observation fusion (linear layer and LayerNorm)	67,712
TTT projections, read LayerNorm, and inner step size	24,705
Conditional diffusion U-Net	65,755,022
Total learned slow parameters	**77,023,951**

The count includes the complete visual feature encoder used by the policy; the unused ResNet classification head is removed. The TTT subtotal consists of 24,576 key, value, and query projection parameters, 128 read-normalization parameters, and one parameter for the positive inner step size.

The 64×64 fast-weight matrix contributes an additional 4,096 state elements per active episode. It is initialized to zero and updated by the inner objective, rather than stored as an independently learned slow parameter. Its initializer is fixed, not learned. History caches, optimizer state, EMA copies, and non-parameter buffers are also excluded from the parameter count. The reported size is for one policy, not the sum over the nine task-specific policies, and is unchanged by the history settings (s, K) or action-gradient window G.

B.2 Online Memory Updates and State

This subsection describes the per-observation update and the state retained for future observations. The same update–read–act sequence runs during training and deployment.

Algorithm 1 gives one online observation step. Both loss terms are differentiated at the same incoming fast weights. The action condition is computed once, after the memory update, and reused across all diffusion steps. During training, the same memory procedure supplies the condition for the noise-prediction loss; the outer gradient boundaries are described below.

Table 5: **Training configuration.** The optimizer, schedule, and EMA settings follow the RMBench DP defaults. The remaining entries specify our sequence-training and TTT setup.

Setting	Value
Optimizer	AdamW (Loshchilov & Hutter, 2019)
Learning rate / weight decay	10^{-4} / 10^{-6}
Adam moments / epsilon	$(0.95, 0.999)$ / 10^{-8}
Learning-rate schedule	Cosine decay with 500 warmup updates
EMA	Inverse gamma 1.0, power 0.75, maximum decay 0.9999
Training / validation data	45 / 5 demonstrations per task
Training duration	20,000 optimizer updates; EMA checkpoint for evaluation
Gradient clipping	Global norm 1.0
Sequence chunk length	64 observations; state persists across chunks
Batch $\times$ accumulated chunks	32×1; 32 chunks per optimizer update
Observation / memory width	64
Fast-weight initialization	Zero matrix at each episode start
Inner step size η	Learned positive scalar, initialized to 0.1
Temporal weight / temperature	$\lambda = 0.3$ / $\tau = 0.1$
Normalization floor	$\epsilon = 0.05$
Reference history / gradient window	$s = 16$, $K = 16$, $G = 1$; varied as stated in each comparison

State and cache storage. The implementation keeps up to sK past values to select ages $s, 2s, \ldots, Ks$, and up to s keys for the historical input. Its main online storage is $O(d_v d_k + sKd_v + sd_k)$, in addition to masks and per-step vectors. At $s = K = 16$ and $d_k = d_v = 64$, this includes 4,096 fast-weight entries, 16,384 cached value entries, and 1,024 cached key entries. The fast-weight count alone is not the total memory footprint. Deployment fixes the slow weights while continuing these memory and cache updates.

B.3 Training Protocol

We follow the default RMBench DP optimizer, learning-rate schedule, diffusion configuration, and EMA settings (Chen et al., 2026). We adapt the observation conditioning and training loop to process episodes with persistent TTT state. Table 5 summarizes the common configuration and the sequence-training settings used here.

Data and checkpoints. Each task uses 45 training demonstrations and five validation demonstrations. Policies are trained separately for each task. Reported evaluations use exponential-moving-average checkpoints after 20,000 optimizer updates.

Action supervision. Following the denoising noise-prediction objective (Ho et al., 2020), the action loss is the squared error between the predicted and sampled noise, averaged over valid action entries. It trains the encoder, projections, read normalization, action head, and positive inner step size η through the memory update. The same η scales both terms in Equation 5. The reference TC–TTT setting uses $\lambda = 0.3$, $\tau = 0.1$, and normalization floor $\epsilon = 0.05$.

Sequence training and gradient boundaries. Training follows each episode in contiguous chunks of 64 observations. The fast weights and detached key/value caches carry across chunk and optimizer-step boundaries; an episode-start mask resets the state for that episode lane. Cached representations are not recomputed after a slow-weight optimizer step. Memory and history caches persist across 64-observation chunks and are reset at episode boundaries. The action loss differentiates through the current read and update; after at most G updates, the fast weights are detached before being carried forward. Cached keys and values are detached when stored, independently of this gradient boundary. Figure 2 summarizes training and inference.

B.4 Evaluation Protocol

Both $M(1)$ and $M(n)$ tasks use an evaluation budget of 100 rollouts per task. This protocol applies to the primary comparisons in Tables 2 and 7, the TC–TTT benchmark comparison, and the history-sampling study in Figure 3. Success follows the RMBench task criteria (Chen et al., 2026). Compared policies draw from common scene-seed manifests. We compute success rates over completed rollouts per task and average them with equal task weights before rounding. Task scores are shown as integers, and success means retain one decimal place. With all 100 rollouts complete, each additional success contributes one percentage point to the corresponding task score. Table 7 reports the per-task results.

C SUPPLEMENTARY EXPERIMENTAL RESULTS

This appendix provides the RMBench comparison and the per-task results for Q1–Q3. Implementation and protocols are specified in Appendix B.

C.1 RMBENCH POLICY COMPARISON

Table 6 retains every policy in Table 1, including MemoAct's four-task results, and adds policies with complete nine-task results. Policies are grouped by policy/backbone scale (< 1B or ≥ 1B), with increasing reported mean success within each group.

Table 6: **RMBench policy comparison.** Success rates (%), grouped by policy/backbone scale and sorted by increasing mean within each group. MemoAct reports four tasks; all other means cover nine tasks. Means over different task subsets are not directly comparable.

Policy	Obs.	Put	Rear.	S-Blk.	S-T	Batt.	Rank	Cov.	Press	Mean
Policy/backbone < 1B										
DP[a]	1	0	0	11	20	10	10	0	0	5.8
ACT[a]	1	0	29	2	2	19	0	0	0	5.9
LiLa-WAM[i]	10	14	20	10	18	24	30	0	0	14.0
TTT ($G = 16$, ours)	2	88	100	24	75	38	36	3	0	40.7
MemoAct[j]	4	41	98	–	53	–	–	–	–	49.1 ± 2.4
AR-K[i]	44	58	62	68	18	38	18	72	82	51.1
TC–TTT (ours)	4	85	98	80	81	37	55	19	7	51.8
AR-Q[i]	84	88	86	82	16	52	34	78	90	67.8
Policy/backbone ≥ 1B										
Fast-WAM[i]	0	0	0	0	7	20	26	0	0	5.9
$\pi_{0.5}$[a]	9	11	13	24	15	16	6	0	0	10.4
HiMem-WAM[h]	28	32	33	38	27	28	24	19	8	26.3
MemoryVLA[b]	2	81	53	76	9	33	53	69	0	41.7
Mem-0[a]	4	90	89	67	14	28	18	68	0	42.0
EventVLA[b]	21	95	96	96	87	35	81	97	3	67.8
DiM-WAM[f]	13	98	99	96	97	48	87	56	34	69.8
LingBot-VA[c]	13	100	100	99	88	41	100	79	84	78.2
MemoryWAM[c]	27	100	100	100	94	41	100	98	87	83.0
MaP-WAM[g]	19	100	66	97	96	82	94	100	96	83.3
SimpleMemVLA[e]	65	100	100	100	93	90	100	98	100	94.0
BridgeVLA++[d]	81	100	100	99	96	96	100	99	93	96.0

Sources. [a]Chen et al. (2026), Table 1; [b]Yang et al. (2026b), Table 8; [c]Yang et al. (2026c), Table 1; [d]Li et al. (2026), Table II; [e]Yin et al. (2026), Table 1; [f]Wang et al. (2026), Table II; [g]Zhao et al. (2026), Table 1; [h]Sun et al. (2026), Table 1; [i]Jiang et al. (2026a), Table IV; [j]Tan et al. (2026). Source-reported means and deviations are retained.

Size groups. Groups reflect policy/backbone scale, not full deployment cost. Fast-WAM and LiLa-WAM follow Yuan et al. (2026); Yang et al. (2026a); DP/ACT follow the official RMBench architectures, and MemoAct follows its official model statistics. AR-K/AR-Q exclude their external agents; Mem-0 adds a planner on $M(n)$ tasks.

Coverage. Dashes denote unreported task scores, not zeros. MemoAct is retained from Table 1 despite its partial coverage. Source protocols differ; for example, source i uses 50 trials per task.;

C.2 PER-TASK RESULTS FOR MAIN-TEXT EXPERIMENTS

Table 7 expands Q1's primary comparisons, Q2's history-sampling figure, and Q3's current-write replacements. Only experiments used in the main text are included.

Table 7: **Per-task results for the main-text experiments.** Panels (a–b) report success (%); panels (c–e) report relative action-MSE changes (%, lower is better). Task-level values are rounded to integers. Means use unrounded values with equal task weights, retaining one decimal place in (a–b) and two in (c–e). TC denotes TC–TTT. Only configurations used in the main text are listed.

Experiment / configuration	Obs.	Put	Rear.	S-Blk.	S-T	Batt.	Rank	Cov.	Press	Mean
(a) Q1: primary comparisons; TC–TTT uses $G = 1$										
TTT, $G = 1$	3	42	98	18	46	20	52	7	0	31.8
TTT, $G = 16$	2	88	100	24	75	38	36	3	0	40.7
TC, $K = 16, s = 1$	0	83	90	82	60	31	53	2	3	44.9
TC, $K = 16, s = 8$	3	86	100	89	37	36	49	16	5	46.8
TC, $K = 16, s = 16$	4	85	98	80	81	37	55	19	7	51.8
TTT, $G = 32$	0	84	99	23	57	41	39	4	2	38.8
TTT, $G = 64$	1	76	100	26	64	33	38	1	0	37.7
TC, $s = 1, K = 32$	1	94	98	83	55	42	52	15	2	49.1
TC, $s = 1, K = 64$	2	61	98	44	45	33	52	14	8	39.7
(b) Q2: history sampling at $G = 1$, targeting 100 rollouts per task										
$s = 1, K = 8$	0	81	74	32	35	12	19	2	3	28.7
$s = 8, K = 8$	5	100	95	93	33	16	22	9	1	41.6
$s = 16, K = 8$	8	100	95	66	74	25	36	12	8	47.1
$s = 1, K = 16$	0	83	90	82	60	31	53	2	3	44.9
$s = 8, K = 16$	3	86	100	89	37	36	49	16	5	46.8
$s = 16, K = 16$	4	85	98	80	81	37	55	19	7	51.8
$s = 1, K = 32$	1	94	98	83	55	42	52	15	2	49.1
$s = 8, K = 32$	3	75	100	82	74	22	44	12	4	46.2
$s = 1, K = 64$	2	61	98	44	45	33	52	14	8	39.7
(c) Q3: current-write replacements, $s = 1$, $K = 16$, $G = 1$										
TC write removed	1	124	37	11	25	15	23	9	25	29.85
Local slope (9 frames)	1	3	8	−1	−4	2	7	7	1	2.60
Full-window slope	0	31	−7	−2	8	23	2	5	1	6.85
Local direction, window norm	0	27	18	−2	4	3	2	−4	2	5.53
Window endpoint secant	0	32	−1	0	9	12	2	2	1	6.29

Continued on next page

Table 7 continued

Experiment / configuration	Obs.	Put	Rear.	S-Blk.	S-T	Batt.	Rank	Cov.	Press	Mean
(d) Q3: current-write replacements, $s = 8$**,** $K = 16$**,** $G = 1$										
TC write removed	−2	399	72	39	47	78	52	39	319	115.79
Local slope (9 frames)	6	1000	577	86	18	62	154	191	1313	378.63
Full-window slope	−1	689	541	44	109	78	4	60	108	181.17
Local direction, window norm	0	318	255	4	21	28	13	15	72	80.74
Window endpoint secant	−1	428	99	28	91	56	9	39	68	90.82
(e) Q3: current-write replacements, $s = 16$**,** $K = 16$**,** $G = 1$										
TC write removed	−1	248	55	11	46	122	294	75	273	124.58
Local slope (9 frames)	3	804	68	55	19	131	210	901	1097	365.37
Full-window slope	0	245	26	8	30	41	19	71	76	57.35
Local direction, window norm	−1	84	7	5	14	52	15	30	309	57.17
Window endpoint secant	0	181	19	5	25	28	13	53	25	38.81

Success results. Panels (a–b), Tables 1 and 2, and Figure 3 use the same completed-rollout results for shared configurations. Each task score is the fraction of successes among completed rollouts. We use an evaluation budget of 100 rollouts per task.

Write replacements. Panels (c–e) expand Table 3, using all nine tasks and one EMA checkpoint per task at each setting. Candidate stride equals s. For each task, we average normalized expert-action MSE over five validation trajectories, 12 sampled times per trajectory, and two paired action samples from the 100-step DDPM. Each entry is $100(E_{\text{replacement}}/E_{\text{native}} - 1)$; the native write is zero by definition. Means are computed before rounding. The checkpoints, incoming memory, reconstruction update, NCE scaling, normalization Jacobian, and read/write addresses stay fixed. Each next observation resumes the native memory trajectory. The local and full-window slopes are OLS fits to normalized value features; the local fit uses nine frames (9f). The full window runs from the oldest available historical candidate to the current frame; the secant uses its endpoints. Slopes and secants are scaled by the weighted historical age. The norm-matched control rescales the local write to the full-window write's norm. These are offline action-prediction errors, not closed-loop success rates.