ReCAT: Remember, Count, and Time: Structured Recurrent Memory for Robot Manipulation

Pankhuri Vanjani[1,*], Mostafa Hatab[1,*], Can Mizrakli[1,*], Vaisakh Shaj[2], Zhuoyue Li[1], Moritz Reuss[3], Rudolf Lioutikov[1,4]

[1]Intuitive Robots Lab (IRL), Karlsruhe Institute of Technology (KIT), Germany [2]University of Edinburgh
[3]NVIDIA [4]Robotics Institute Germany (RIG)
[*]Core contributors

Abstract: Memory-dependent manipulation requires robots to make decisions using information that is no longer available to their current sensors, such as recalling an earlier visual cue, tracking task progress, counting repeated events, or estimating elapsed time. We present ReCAT, a language-conditioned policy with structured recurrent memory. An instruction-conditioned encoder forms features from the current observation. A recurrent memory integrates the observation stream through Mamba-2 layers and one causal attention layer. A flow-matching Transformer decoder reads the current and the historical representation through separate cross-attention in every block. ReCAT reaches 95.3% average success on LIBERO and 62.4% on RMBench, with the best or tied-best result on six of nine tasks. On three real-robot tasks probing spatial recall, event counting, and interval timing, the best ReCAT variant reaches 66.7% average success, against 8.3% for the strongest short-history baseline. Controlled comparisons within ReCAT show that the observation encoder and every-block memory conditioning are needed for this performance.

Keywords: Robot Manipulation, Recurrent Memory, Imitation Learning

1 INTRODUCTION

Robot manipulation often requires information that is no longer available in the current observation. An object's original location may have left the camera view, the number of completed repetitions may not be apparent, or the next action may depend on how much time has elapsed. In the PLANT task (Fig. 1), for example, the shovel-over-pot view looks alike after one scoop and after two, yet the required actions differ: scoop again, or put down the shovel and proceed to planting. Such tasks are non-Markovian with respect to the current observation, and the policy must combine current perceptual information with relevant interaction. Policies obtain this history in different ways. Some use a window of recent observations. Some store selected observations in a memory bank [1, 2]. Others summarize the whole observation stream in a compact recurrent state [3, 4, 5]. Recent and concurrent work shows that such full-history recurrence helps in manipulation [6, 7, 8, 9, 10, 11]. Yet, precise manipulation also requires the current spatial and proprioceptive observation. So, a policy needs both a history representation and direct access to the present. We introduce ReCAT,

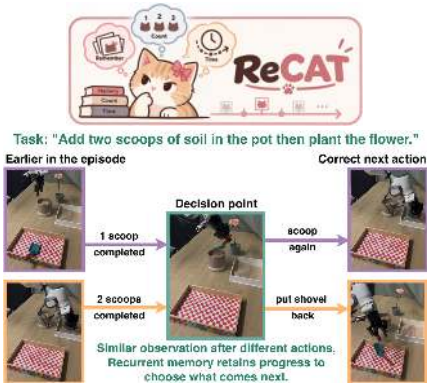


Figure 1: Visually similar observations can require different actions depending on task history. ReCAT retains the completed scoop count in a compact recurrent memory, allowing the robot to scoop again after one scoop and put down the shovel after two.

10th Conference on Robot Learning (CoRL 2026), Austin TX, USA.

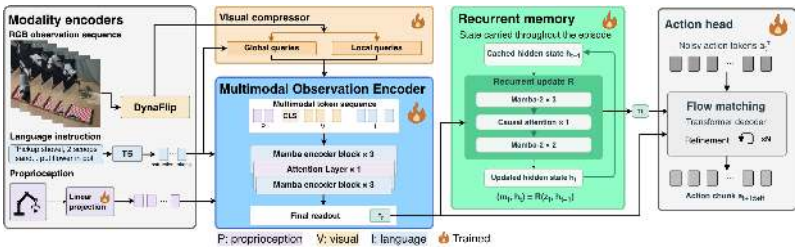


Figure 2: Frozen DynaFLIP and T5 encoders (LoRA-adapted) feed a language-conditioned visual compressor and a six-layer Mamba–attention frame encoder, which produces one feature vector z_t per step. The same z_t feeds the action decoder directly and is the input to the recurrent memory, which returns the readout m_t. The flow-matching decoder attends to z_t and m_t through separate cross-attention in every block. The decoder backbone is shared across all experiments.

a structured memory policy for language-conditioned manipulation. We evaluate ReCAT on LIBERO, RMBench, and three real-robot tasks. The real-robot tasks probe spatial recall, event counting, and interval timing. ReCAT reaches 95.3% average success on LIBERO and 62.4% on RMBench. On the real-robot tasks it reaches 66.7%, compared with 8.3% for the strongest short-history baseline. Stage-wise evaluation shows where a policy fails: during manipulation, or when it must use history. Controlled comparisons within ReCAT then show how observation encoding, recurrent update rule, memory capacity, and decoder conditioning affect performance. Mamba variants reach the highest success on counting and timing tasks.

2 METHODOLOGY

2.1 Problem Formulation

We study language-conditioned imitation learning from demonstrations $\mathcal{D} = \{(\ell^{(i)}, (s_t^{(i)}, a_t^{(i)})_{t=1}^{T_i})\}_{i=1}^{N}$, where $s_t = (v_t, p_t)$ contains visual and proprioceptive observations. Let $\mathcal{H}_t = (s_{1:t}, \ell)$ denote the observation history and $A_t = (a_t, \ldots, a_{t+K-1})$ an action chunk. Because the current bservation may not reveal task state in memory-dependent tasks, we model the policy as $\pi_\theta(A_t \mid \mathcal{H}_t)$. ReCAT summarizes this history in a compact recurrent state updated as observations arrive.

2.2 ReCAT

ReCAT has three parts (Fig. 2). An instruction-conditioned observation encoder turns the current images, instruction, and proprioception into one feature vector z_t. A temporal memory takes z_t as its input at every step, updates its retained state h_t, and returns a history-conditioned readout m_t. A flow-matching action decoder predicts the next action chunk from z_t and m_t. The same z_t therefore feeds the decoder directly and drives the memory.

Current-Observation Encoding DynaFLIP [12] encodes each camera image into a 16×16 patch grid P_t. Its T5 text encoder produces instruction tokens L. Both backbones stay frozen and are adapted with LoRA. Following Compressor-VLA [13], we compress each patch grid with language-conditioned global queries and local window queries. The global queries keep task-relevant context. The local queries keep fine spatial detail.

$$Z_t = \mathrm{CrossAttn}\big(\gamma(\bar{\ell}) \odot Q + \beta(\bar{\ell}),\, P_t\big), \tag{1}$$

Here $\bar{\ell}$ summarizes the instruction, Q are the learned global queries, and γ, β are FiLM parameters. The local window queries work the same way over their windows. Global and local outputs are concatenated into 80 tokens per camera. A six-layer encoder then mixes these tokens with the instruction tokens and a proprioception token within the frame. Five of its layers are Mamba-2 and the fourth is bidirectional attention. This encoder carries no state across timesteps. Its last position is read out as the frame feature $z_t \in \mathbb{R}^{768}$.

Recurrent Memory The recurrent memory summarizes the whole observation history. It is a stack of six blocks of width 768. Five blocks are Mamba-2 layers and have no feed-forward network (FFN). The fourth block is a causal attention layer and has one FFN. At every step the memory takes z_t as input, updates its retained state, and returns a readout,

$$(q_t, m_t) = R_\phi(z_t, q_{t-1}), \tag{2}$$

where $q_t = (h_t, C_t)$. Here h_t denotes the persistent recurrent states of the five Mamba-2 layers, and C_t is the key-value cache of the attention layer. The size of h_t is fixed by the architecture. The cache C_t holds one entry per step and therefore grows linearly with episode length. Each entry is one 768-dimensional vector, so the memory never stores images or patch tokens. All of q_t is reset at the start of each episode. The readout m_t is the normalized output of the last block, not the state itself. We vary memory capacity (number of blocks and Mamba-2 state width d_{state}) and recurrent update rule. For the latter, we replace the default Mamba-2 layers with Mamba-3 or Gated DeltaNet-2, keeping all other components and the training recipe fixed. Training uses a parallel scan over each trajectory, while inference updates the state one step at a time.

Observation and memory conditioned Action Generation Following flow-matching policies [14], a Transformer decoder predicts an action chunk $A_t \in \mathbb{R}^{K \times d_a}$ with $K = 16$. The decoder has four blocks. Each block applies self-attention over the action tokens, then cross-attention to z_t, then cross-attention to m_t, then a feed-forward network. The two cross-attention operations have separate parameters, so the current observation and the memory enter every block through separate paths. Flow time τ conditions every block through zero-initialized adaptive layer normalization.

3 Evaluation

We evaluate ReCAT on two simulation benchmarks and three real-robot tasks. The evaluation asks three questions: **(RQ1)** How effective is ReCAT on standard and on memory-dependent manipulation? **(RQ2)** How do its observation encoder and its memory conditioning affect performance?**(RQ3)** How do the recurrent update rule and the memory capacity affect different memory demands? Unless stated otherwise, ReCAT denotes the Mamba-2 instantiation. Further baseline and evaluation details, stage-wise results, and computational costs are provided in the Appendix.

RQ1: ReCAT is competitive on standard manipulation and strong on memory-dependent manipulation. On LIBERO, ReCAT reaches 95.3% average success across the four suites (Table 2). Diffusion Policy reaches 72.4%. X-VLA, which uses large-scale pretraining, reaches 98.1%. On LIBERO-Plus, ReCAT reaches 64.2% and X-VLA 71.4%. ReCAT therefore remains competitive on tasks that the current observation alone can solve. On RMBench, ReCAT reaches 62.4% average success over nine tasks (Table 3). It has the best or tied-best score on six of them, including three of the four $M(n)$ tasks. EventVLA has a higher overall average of 67.9%. It builds on a 4B-parameter backbone, about ten times larger than ReCAT, with an additional large-scale pretraining phase. Performance varies across tasks. ReCAT reaches 100% on Put Back Block and Swap Blocks but 4% on Observe and Pick Up and 6% on Press Button.

The real-robot tasks show the largest gap (Table 1). Policies without persistent memory succeed in 0-8.3% of rollouts. X-VLA-H, the strongest baseline, reaches 8.3% (95% Wilson CI [3.6%, 18.1%]). ReCAT variants reach 50.0-66.7%, and the best variant reaches 66.7% (95% CI [54.1%, 77.3%]). The two intervals do not overlap. All three ReCAT variants wait the correct interval in every POT TIMER rollout, and the Mamba variants complete 85-90% of PLANT rollouts. The stage-wise columns of Table 1 show where the baselines fail, which we examine next. Across tasks, baselines usually complete the initial manipulation but fail at the history-dependent decision, whereas ReCAT typically reaches that decision and succeeds.

RQ2: The observation encoder and memory conditioning both affect performance, especially on PLANT. *Observation encoder.* With Mamba-2 memory fixed, the hybrid Mamba-attention encoder achieves 66.7% average success, versus 25.0% for the full Transformer (Table 4); the largest gap is on

Table 1: Stage-wise real-robot results. Each task progresses from observation-driven stages to a history-dependent decision (Fig. 3). For PLANT, scoop columns are mutually exclusive and count scoops before proceeding to the flower or timing out. For POT TIMER, "No wait" and "Wrong wait" indicate leaving the waiting state early or outside the 29–31 s window. For SPONGE, return outcomes partition rollouts that placed the sponge, excluding those that stalled at the plate. Success is the final column for each task. Values are percentages over 20 rollouts with randomized start positions.

	Plant				Pot timer						Sponge			
Method	1 scoop	2 scoops	3 or more	Flower planted Success	Pot on stove	No wait	Wrong wait	≈30 s wait	Pepper in the pot	Success	Pick sponge, put on plate	Returned to wrong position	Returned to same position Success	Avg.
Baselines														
Diffusion policy [15]	–	–	–	0	–	–	–	–	–	0	–	–	0	0.0
Diffusion policy [15] + history	15	10	0	0	100	25	0	0	25	0	15	–	0	0.0
X-VLA [16]	–	–	–	0	–	–	–	–	–	0	–	–	0	0.0
X-VLA [16] + history	0	10	85	10	100	0	35	5	40	5	75	65	10	8.3
Gated Memory Policy (GMP) [17]	65	20	10	15	95	20	10	5	35	5	50	50	0	6.7
Ours														
ReCAT-Mamba-2	0	100	0	85	100	0	0	100	95	95	20	0	20	**66.7**
ReCAT-Mamba-3	0	95	0	**90**	100	0	0	100	100	**100**	0	0	0	63.3
ReCAT-Gated DeltaNet 2	0	70	0	20	100	0	0	100	90	90	40	0	**40**	50.0

PLANT (85% vs. 0%). The pure-SSM variant reaches 25% on PLANT. Removing either the memory module or multimodal encoder reduces success to 0% across all three tasks. *Memory conditioning.* Cross-attention to memory in every decoder block achieves 66.7% average success, compared with 38.3% for late fusion and at most 31.7% for other alternatives (Table 5). The largest gap is on PLANT (85% vs. at most 10%). POT TIMER is less sensitive: variants that place the pot generally wait correctly. On SPONGE, late fusion reaches 25% success versus 20% for every-block attention, a difference of one rollout; no variant achieves high success.

RQ3: Update rule and memory capacity have task-dependent effects. *Update rule.* With the encoder, attention layer, and decoder fixed, we vary the recurrent update (Table 1). On PLANT, Mamba-2 and Mamba-3 achieve 85% and 90% success, compared with 20% for Gated DeltaNet-2. All three rules wait the correct interval in every successful POT TIMER rollout. On SPONGE, Gated DeltaNet-2 achieves the highest success (40%), followed by Mamba-2 (20%) and Mamba-3(0%). These results suggest that update rules interact with task demands. Additive writes may support counting by accumulating repeated evidence, while delta updates may represent multiplicity less directly. This interpretation remains a hypothesis because we do not measure the internal count. Mamba-2 has the highest average success and is used in the remaining experiments. *Memory capacity.* Reducing memory depth from six to two blocks lowers PLANT success from 85% to 65%. Varying $d_{\text{state}} \in \{32, 64, 128, 256\}$ yields success rates of 15%, 20%, 85%, and 0%, respectively. The non-monotonic results indicate sensitivity to capacity under a single training run, rather than that larger states necessarily perform worse. We use six blocks and $d_{\text{state}} = 128$ in the remaining experiments.

4 CONCLUSIONS

We presented ReCAT, a structured recurrent policy for memory-dependent manipulation. Its memory integrates observation history through Mamba-2 layers and causal attention, while its flow-matching decoder separately attends to current observations and memory. ReCAT achieves 95.3% on LIBERO, 62.4% on RMBench, and 66.7% across three real-robot tasks. Its inference latency is 59 ms per step, the lowest among compared memory-based policies. Stage-wise results show that short-history policies often fail at decisions requiring earlier information. Among evaluated configurations, the hybrid encoder and per-block memory cross-attention achieve the highest average success. Update rules have task-dependent effects: additive rules perform best on counting and timing, while the delta rule performs best on spatial recall. Effective memory-dependent control depends on what is written, how it is updated, and how the action generator reads it alongside current observations.

Our real-robot evaluation covers one task per memory demand and 20 rollouts per task. Broader studies are needed to assess generalization across tasks, embodiments, counts, and durations. SPONGE highlights the challenge of combining memory with fine-grained spatial control which would be our future work.

References

[1] H. Shi, B. Xie, Y. Liu, L. Sun, F. Liu, T. Wang, E. Zhou, H. Fan, X. Zhang, and G. Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. In *International Conference on Learning Representations*, volume 2026, pages 18567–18602, 2026.

[2] H. Li, F. Shen, D. Chen, L. Yang, X. Wang, J. Shi, Z. Bing, Z. Liu, and A. Knoll. Remem-vla: Empowering vision-language-action model with memory via dual-level recurrent queries. *arXiv preprint arXiv:2603.12942*, 2026.

[3] A. Gu, K. Goel, and C. Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2021.

[4] S. Yang, J. Kautz, and A. Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. *arXiv preprint arXiv:2412.06464*, 2024.

[5] A. S. Lahoti, K. Li, B. Chen, C. Wang, A. Bick, Z. Kolter, T. Dao, and A. Gu. Mamba-3: Improved sequence modeling using state space principles. In *International Conference on Learning Representations*, volume 2026, pages 86173–86202, 2026.

[6] Y. Zhou, Y. Lin, F. Peng, J. Chen, K. Huang, H. Yang, and Z. Yin. Mtil: Encoding full history with mamba for temporal imitation learning. *IEEE Robotics and Automation Letters*, 2025.

[7] Y. Yoo, J. Hu, Y. Zhu, B. Liu, Q. Liu, R. Martín-Martín, and P. Stone. Robossm: Scalable in-context imitation learning via state-space models. *arXiv preprint arXiv:2509.19658*, 2025.

[8] T. Tsuji. Mamba as a motion encoder for robotic imitation learning. *IEEE Access*, 13:69941–69949, 2025.

[9] Z. Guan, J. Hu, H. Fang, Y. Jiang, Y. Huang, S. Li, X. Li, and Y. Ban. Dssp: Diffusion state space policy with full-history encoding. *arXiv preprint arXiv:2605.14598*, 2026.

[10] Y. Zhou, Y. Wang, N. Wang, S. Xing, S. Tu, X. Li, J. Zhang, N. Jiang, Y. Lin, H. Yang, et al. Chronos: A physics-informed full-history framework for non-markovian long-horizon manipulation. *arXiv preprint arXiv:2606.30318*, 2026.

[11] E. Cherepanov, N. Kachaev, D. Zelezetsky, A. Bulatov, A. Pshenitsyn, Y. Kuratov, A. Skrynnik, A. I. Panov, and A. K. Kovalev. μvla: On recurrent memory for partially observable manipulation in vla models. *arXiv preprint arXiv:2606.12497*, 2026.

[12] J. Lee, S. Lee, J. Shin, H. Jung, S. Kim, D. Cho, H. J. Kim, J.-B. Huang, and F. Huang. Dynaflip: Rethinking robotics perception via tri-modal-dynamics guided representation. *arXiv preprint arXiv:2605.30350*, 2026.

[13] J. Gao, F. Ye, J. Zhang, and W. Qian. Compressor-vla: Instruction-guided visual token compression for efficient robotic manipulation. *arXiv preprint arXiv:2511.18950*, 2025.

[14] M. Reuss, H. Zhou, M. Rühle, Ömer Erdinç Yağmurlu, F. Otto, and R. Lioutikov. Flower: Democratizing generalist robot policies with efficient vision-language-action flow policies, 2025. URL https://arxiv.org/abs/2509.04996.

[15] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023.

[16] J. Zheng, J. Li, Z. Wang, D. Liu, X. Kang, Y. Feng, Y. Zheng, J. Zou, Y. Chen, J. Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *International Conference on Learning Representations*, volume 2026, pages 60580–60606, 2026.

[17] Y. Gao, J. Liu, S. Li, and S. Song. Gated memory policy. *arXiv preprint arXiv:2604.18933*, 2026.

[18] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36: 44776–44791, 2023.

[19] S. Fei, S. Wang, J. Shi, Z. Dai, J. Cai, P. Qian, L. Ji, X. He, S. Zhang, Z. Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

[20] T. Chen, Y. Wang, M. Li, Y. Qin, H. Shi, Z. Li, Y. Hu, Y. Zhang, K. Wang, Y. Chen, et al. Rmbench: Memory-dependent robotic manipulation benchmark with insights into policy design. *arXiv preprint arXiv:2603.01229*, 2026.

[21] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[22] A. J. Scholp, M. R. Hoffman, S. P. Rosen, S. M. Abdelhalim, C. A. Jones, J. J. Jiang, and T. M. McCulloch. Spectral arc length as a method to quantify pharyngeal high-resolution manometric curve smoothness. *Neurogastroenterology & Motility*, 33(10):e14122, 2021.

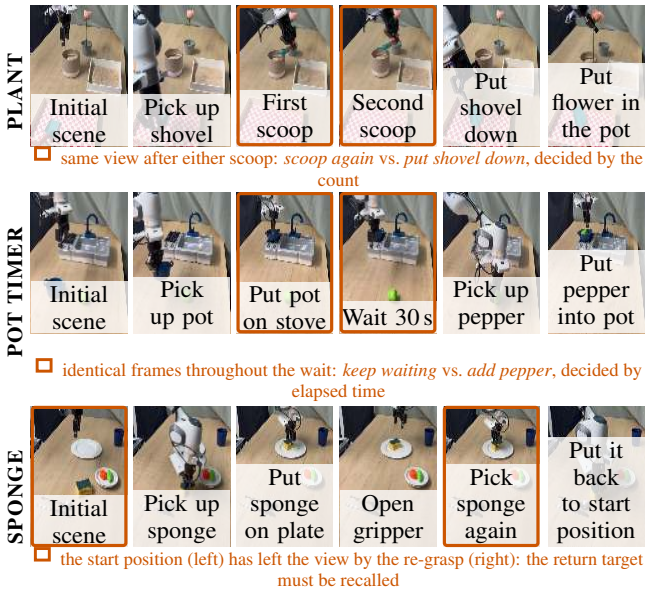


Figure 3: Real-robot tasks, one per row, each isolating one memory demand: PLANT counts scoops across five stages, POT TIMER times an interval while the scene is static, and SPONGE recalls an unmarked start position that has left the camera view. Outlined frames mark the decision point where the correct action depends on an earlier event, not on the current observation.

A Appendix

A.1 Experimental setup

Simulation benchmarks We evaluate ReCAT on LIBERO [18], including Spatial, Object, Goal, and LIBERO-10, using two cameras and 50 epochs. Each 10-task suite is evaluated over 500 episodes (50 per task). Since LIBERO is largely Markovian, it tests whether full-history recurrence preserves performance when the current observation is sufficient. We use LIBERO-Plus [19] to evaluate robustness to controlled visual and physical perturbations. RMBench [20] contains nine long-horizon bimanual tasks: five $M(1)$ and four $M(n)$ in which actions depend on past events no longer observable. Following its protocol, we train one policy per task on 50 demonstrations and evaluate 100 rollouts with unseen scenes.

Real robot setup We further evaluate the policies on three real-robot tasks that probe distinct memory demands under real-world execution variability. Unlike simulation, real-world manipulation introduces visual clutter, contact variation, friction, and trajectory variability, which can interfere with retaining task-relevant history. We therefore consider three tasks requiring *spatial recall*, *discrete event counting*, and *interval timing*, each containing states where the correct action cannot be determined from the current observation alone. The three tasks are shown in Fig. 3. We use a single-arm Franka Emika Panda in the DROID setup [21]. The robot receives 224×224 RGB images from a fixed right-side camera and a wrist-mounted camera, together with joint-space proprioception. Demonstrations are collected through teleoperation at 15 Hz. The policy predicts an eight-dimensional action comprising seven joint positions and a continuous gripper width.

The **SPONGE** task evaluates spatial recall under clutter and broad spatial variation. The sponge is initialized across the robot's reachable workspace without any positional markers indicating its start location. The robot places it on a plate, releases it, and must then return it to its original position, which is no longer directly observable once the sponge has moved. This requires the remembered location to be maintained through the intervening manipulation and surrounding clutter. 48 demonstrations, $\sim$480 steps per episode.

The **PLANT** task evaluates discrete event counting within a long-horizon, multi-stage manipulation sequence. The robot must place exactly two scoops of soil into a pot, put down the shovel, and pick up and insert a flower into the pot. Since the observations during the two scooping cycles are visually similar, the policy must track the number of completed scoops to determine whether to repeat the scooping action or proceed to the remaining stages. 45 demonstrations, $\sim$1,200 steps per episode, with varying scoop durations (125.4 ± 24.3 frames).

The **POT TIMER** task evaluates interval timing. The robot places a pot on a stove, waits for approximately 30 seconds, and then picks up a pepper and places it in the pot. The visual scene remains nearly unchanged during the waiting period and provides no direct indication of elapsed time, requiring the policy to maintain temporal state internally. 45 demonstrations, $\sim$980 steps per episode.

Protocol We train one policy per task and configuration with a single seed for 500 epochs and evaluate the final checkpoint. Each policy runs 20 rollouts per task with randomized initial object positions. We report final success and the stage each rollout reached. In SPONGE, a return counts as success when the sponge lands within 2-4 cm of its original centre. In POT TIMER, a wait counts as correct when it lasts 29-31 s. With 20 rollouts, one rollout is five percentage points, and differences of that size are within rollout noise.

Metrics We report *success rate* in all evaluations. For the real robot we also report *policy inference latency* and its reciprocal rate (Table 6), and *trajectory smoothness* as the average spectral arc length (SPARC) [22] of the joint velocity profiles across all joints. Higher SPARC is smoother.

Implementation and training Training processes each demonstration as one causal scan over the full episode. The vision and text backbones stay frozen and are adapted with LoRA of rank 16 and 8, respectively. Real-robot policies train for 500 epochs. At inference, the decoder integrates the velocity field with $N = 4$ Euler steps. The robot executes the first $k = 4$ actions of each chunk and then replans. The frame encoder runs and the memory advances at every control step, including during chunk execution. The memory's recurrent states and attention cache are reset between episodes.

A.2 Baselines

For LIBERO and RMBench we report results from prior work. For the real-robot tasks we retrained every baseline on our demonstrations with the same cameras, control rate, action space, and rollout protocol. The short-history variants are our own extensions, since neither base method ships one. The baselines cover three ways a policy can access the past. All baselines run on the robot's inference GPU 4060Ti. This excludes memory-augmented VLAs built on multi-billion-parameter backbones [1]. An official implementation of DSSP [9] was not available at the time of our experiments, so we discuss it as concurrent work and restrict comparisons to the methods above and to reported benchmark results. The baselines cover three ways a policy can access the past. **(i) No memory: DP** is a standard Diffusion Policy conditioned on the current right- and wrist-view images and proprioception, and **X-VLA** [16] is a vision-language-action policy with no persistent temporal memory. **(ii) Short-term memory: DP-H** and **X-VLA-H** add the last eight observations (stride 10), encoded by a shared trainable ResNet-18 and by a frozen Florence-2 encoder compressing them into eight history tokens through a two-layer Transformer, respectively. **(iii) Explicit attention-based memory: Gated Memory Policy (GMP)** [17] retrieves visual and action information from a finite stored history using cross-attention and a calibrated binary memory gate. Diffusion Policy is the standard imitation-learning policy at a size comparable to ours (268M parameters). X-VLA is a 0.9B-parameter pretrained VLA and one of the strongest model that fits the robot's GPU. GMP is the closest published memory policy at comparable size. All baselines were selected to be deployable on the robot's inference GPU, which excludes recent memory-augmented VLAs built on multi-billion-parameter backbones [1].

Table 2: Success rates (%) on LIBERO.

Method	Spatial	Object	Goal	LIBERO-10	Avg.	LIBERO-Plus
Diffusion policy	78.3	92.5	68.3	50.5	72.4	–
X-VLA [16]	**98.2**	**98.6**	**97.8**	**97.6**	**98.1**	**71.4**
Ours	96.4	97.2	97.2	90.4	95.3	64.2

Table 3: Success rates (%) on RMBench. * Indicates models utilizing large scale pretraining.

Task	TMC	DP (90M)	ACT (80M)	$*pi_{0.5}$ (3B)	*X-VLA (0.9B)	MEM-0 (10B)	EventVLA (4B)	*MemoryVLA (7.3B)	*MemER (10B)	Ours (374M)
Observe and Pick Up	$M(1)$	1	1	9	9	4	**21**	2	7	4
Rearrange Blocks	$M(1)$	0	29	13	13	89	**96**	53	17	**96**
Put Back Block	$M(1)$	0	0	11	18	90	95	81	0	**100**
Swap Blocks	$M(1)$	11	2	24	16	67	96	76	14	**100**
Swap T	$M(1)$	20	2	15	3	14	**87**	9	7	64
Average	$M(1)$	6.4	6.8	14.4	11.8	52.8	**79.0**	44.2	9.0	72.8
Battery Try	$M(n)$	10	19	16	26	28	35	33	27	**43**
Blocks Ranking Try	$M(n)$	10	0	6	1	18	81	53	12	**99**
Cover Blocks	$M(n)$	0	0	0	2	68	**97**	69	40	50
Press Button	$M(n)$	0	0	0	0	0	3	0	0	**6**
Average	$M(n)$	5.0	4.8	5.5	7.3	28.5	**54.0**	38.8	19.8	49.5
Overall average	–	5.8	5.9	10.4	9.8	42.0	**67.9**	41.8	13.8	62.4

A.3 Computational efficiency.

Table 6 compares model size, inference cost, and trajectory smoothness under the same hardware and inference settings. ReCAT has 374M parameters, of which 142M are trainable. Policy inference takes 59 ms per step, against 110 ms for DP-H, 93 ms for GMP, and 288 ms for X-VLA-H. The reciprocal rate is 16.9 Hz. ReCAT also has the highest SPARC, that is the smoothest end-effector trajectories, among the compared policies.

B Additional Tables

Table 4: Real-robot success rates for observation-encoder and component ablations.

Setting	Sponge	Plant	Pot timer	Avg.
Ours, hybrid encoder	**20**	**85**	**95**	**66.7**
Ours, full-transformer encoder	10	0	65	25.0
Ours, pure-SSM encoder	–	25	–	–
Ours, w/o multimodal observation encoder tokens	0	0	0	0.0
Ours, without memory module	0	0	0	0.0

Table 5: Step-wise real-robot success across six memory-to-action conditioning mechanisms. Columns as in Table 1.

	Plant				Pot timer						Sponge			
Integration	1 scoop	2 scoops	3 or more	Flower planted Success	Pot on stove	No wait	Wrong wait	≈30 s wait	Pepper in the pot	Success	Pick sponge, put on plate	Returned to wrong position	Returned to same position	Avg.
ReCAT, cross-attention	0	100	0	**85**	100	0	0	100	95	**95**	20	0	20	**66.7**
Integration variants														
Late fusion	0	0	0	0	90	0	0	90	90	90	70†	40	**25**	38.3
AdaLN	35	15	0	5	100	0	0	100	90	90	25	5	0	31.7
Scale	0	25	0	10	80	0	0	80	75	75	40	40	0	28.3
Dropout	5	10	0	0	90	0	0	90	60	60	20	10	0	20.0
Gate	35	5	0	5	50	0	0	50	20	20	20	0	0	8.3

Table 6: Computational cost on the same hardware and inference setting at. Latency is per policy inference; the rate is its reciprocal. SPARC: higher is smoother.

Metric	ReCAT	DP-H	X-VLA-H	GMP
Total / trainable parameters	374M / 142M	267.9M / 267.9M	881.9M / 881.9M	282.2M / 188.7M
Peak inference VRAM	1.44GB	0.65GB	2.55GB	1.09GB
Inference latency/frequency	0.059s/16.9hz	0.11s/9hz	0.288s/3.47hz	0.093s/10.75hz
Smoothness metric (SPARC [22])	-5.16	-6.19	-6.76	-13.06