KEEPING MEMORY IN-CONTEXT: NOISE-MATCHED TRAINING FOR LONG-HORIZON ROBOT CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon robot control requires acting on information that has left the current observation. Memory-augmented vision–language–action policies address this by storing past observations and retrieving them into the policy's context, and recent work has steadily improved what is stored and how it is retrieved. Such systems nevertheless fall far short of what the same policy achieves when the relevant memory is handed to it directly. We show that much of this gap lies not in the memory but in its use: the memory does reach the policy, but the policy was trained on memory in a form different from the one it receives at deployment, and so it does not read it. We propose ROBIN, which closes the training–deployment mismatch behind this *read-out gap* by training the policy to read the memory it will be given at deployment. ROBIN reaches 42.0 on RoboMME with the same backbone as the benchmark's own memory mechanisms, matching or exceeding the per-task scores of the strongest published configuration on 7 of its 16 tasks, reaches the best published M(1) average on RMBench (83.8), and on two physical arms raises success from 25.0 to 47.5 on a retrieval task and from 36.7 to 66.7 on an occlusion task. Whether a policy uses memory is decided less by how much is stored than by how it was trained to read it.

1 INTRODUCTION

Vision–language–action (VLA) policies map the current observation and a language instruction directly to actions (Brohan et al., 2023a; Octo Model Team, 2024; Kim et al., 2024; Physical Intelligence, 2025; Chi et al., 2023; Zhao et al., 2023), and memory-augmented variants extend them to long-horizon tasks by storing past observations or events and retrieving the relevant entry into the policy's context at decision time (Sridhar et al., 2026; Fang et al., 2025; Shi et al., 2025a; Yang et al., 2026; Bu et al., 2025; Wang et al., 2026) (Figure 1). The challenge is that such tasks turn on information no longer in front of the robot, such as which drawer has been searched or where an object was before it was covered, so all the policy knows about the past arrives through that retrieved context.

Existing systems differ in what they store and how they retrieve it (Dai et al., 2026; Chen et al., 2026), but they are trained, with few exceptions, on demonstrations whose context is annotated from the recorded episode, and is therefore noise-free and piecewise constant, whereas at deployment the context is produced by a learned predictor and inherits its error distribution: subgoals are misclassified, withheld, or imprecisely localized, and revised at every frame. The conditioning variable thus undergoes a distribution shift that the imitation objective never sees: a policy that follows the context wherever it points and one that acts from the current view alone are both consistent with low training loss, so the *read-out gap*, the difference in success between the annotated and the predicted context, is left uncontrolled, and success rate alone cannot tell which of the two policies training produced (Chan et al., 2022b; Singh et al., 2023; Liu et al., 2024b).

ROBIN (**R**ead-**O**ut ro**B**ust **IN**terface training) addresses these two blind spots with two components, shown in Figure 2. Memory enters the policy as before, as a one-line subgoal or a set of selected frames placed next to the current view (Figure 2a); ROBIN changes how the policy is trained on it and which policy is kept. **Noise-matched training** (NMT; Figure 2b) trains the policy on the context

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.

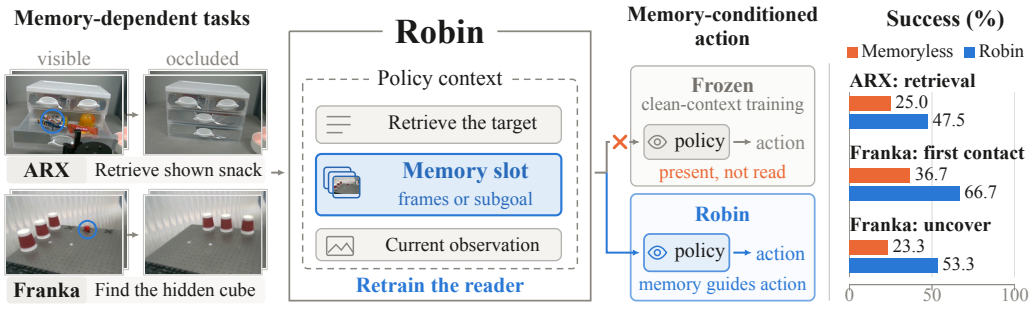

Figure 1: Memory reaches the policy; ROBIN trains the policy to read it. *Left:* two memory-dependent tasks whose target is visible earlier and indistinguishable at execution (ARX snack retrieval, Franka cup-and-cube; frames from our setups). *Center:* stored frames or a predicted subgoal enter the policy's context next to the current observation, and ROBIN retrains the policy on that context. *Right:* success (%) of the memoryless policy and ROBIN on both arms (Table 4); ROBIN's actions follow the context it is given.

it will receive. During training the annotated context passes through a noise operator with three branches, one for each way the deployed predictor fails: it replaces the subgoal with a wrong one, withholds it, or displaces its coordinate by a spatial jitter. The branch probabilities are read from the predictor's error histogram on held-out data, so the error rates the policy trains on are the predictor's own, with no level to tune. Under this operator the context is informative but not authoritative, and the policy learns to weigh it against the current view: it goes where the context points when the scene agrees and falls back on what it sees when the two conflict. When memory arrives as frames rather than as a line of text, the same principle selects the training frames with the rule the policy is deployed with. **Counterfactual read-out probing** (CRP; Figure 2c) makes dependence on memory the criterion for which policy is kept. Holding the observation fixed and substituting the memory, it measures how far the actions move and deploys the policy whose actions depend on that memory rather than the one with the lowest loss; the same test is run on the deployed policy on hardware.

On RoboMME, ROBIN reaches 42.0 against published memory mechanisms built on the same backbone, with the highest Counting-suite average in the table, whose tasks require tracking how many times an event has occurred. On RMBench, a shared nine-task $\pi_{0.5}$ reaches the best published M(1) average (83.8) and 68.1 across the nine tasks, with the gains concentrated on tasks in which the policy must act on an observation made long before. The counterfactual probe shows that the deployed policy's actions respond to a memory swap at 4.6 times the level of the policy trained on the annotated context, and that the response concentrates in the steps where the target must be recalled. On two physical arms whose current view cannot decide the action, success rises from 25.0 to 47.5 on a four-way retrieval task (chance 25%) and first-contact accuracy from 36.7 to 66.7 on a three-position occlusion task.

Our contributions are threefold:

- We propose ROBIN, whose noise-matched training (NMT) trains a memory-conditioned policy under noise matched to the error types and rates its deployed predictor produces, so the policy reads the memory it is given.
- Counterfactual read-out probing (CRP) compares, for the same observation, the actions taken under the true memory and under a substituted one, and keeps the policy whose actions depend on that memory.
- We evaluate NMT and CRP on RMBench and RoboMME, and demonstrate memory-conditioned control on two physical arms in scenes where the current view cannot decide the action.

2 RELATED WORK

VLA policies and world models. Generalist vision–language–action policies map the current observation and an instruction to actions (Brohan et al., 2023b;a; Octo Model Team, 2024; Kim et al., 2024; Black et al., 2024a; Physical Intelligence, 2025; Zheng et al., 2025), built on imitation backbones such as diffusion and transformer policies (Chi et al., 2023; Zhao et al., 2023). Their training

recipes address the gap between demonstration and deployment through on-policy correction (Ross et al., 2011), noise injection (Laskey et al., 2017), scheduled sampling (Bengio et al., 2015), conditioning dropout (Ho & Salimans, 2022), and data diversity (Lin et al., 2025; Lynch et al., 2019), while history inputs are known to invite shortcuts (Haan et al., 2019; Wen et al., 2020). World models extend the same backbones with an explicit model of how the scene evolves (Hafner et al., 2023; Du et al., 2023b; Hu et al., 2024; Wang et al., 2026). ROBIN keeps such a backbone and its imitation objective unchanged, and alters only the form of the memory the policy is trained on.

Memory for robot control. Partial observability is classically handled by recurrent or transformer memory (Ni et al., 2022; 2023; Morad et al., 2023; Pleines et al., 2023; Cherepanov et al., 2025). For robot policies, memory takes the form of retrieval (Du et al., 2023a; Sridhar et al., 2026), memory banks (Fang et al., 2025), dual stores (Shi et al., 2025a), event buffers (Yang et al., 2026), and latent task memory (Bu et al., 2025; Wang et al., 2026), and reaches the policy as language or image subgoals (Ahn et al., 2022; Shi et al., 2025b; Black et al., 2024b) or as coordinate and mark prompts (Yang et al., 2023; Nasiriany et al., 2024; Liu et al., 2024a; Yuan et al., 2024; Zawalski et al., 2024); RoboMME (Dai et al., 2026) and RMBench (Chen et al., 2026) benchmark these mechanisms under a shared backbone. These systems decide what is stored and how it is retrieved, and curricula that anneal the memory a policy is trained with have been applied on that construction side (Yang et al., 2026); ROBIN matches the error distribution of the predictor that writes the context at deployment, and changes how the policy reads the memory already retrieved into it.

In-context learning. Once memory is in the context, using it is in-context learning. Transformers acquire this ability from distributional properties of the data (Chan et al., 2022b; Garg et al., 2022; Olsson et al., 2022; Akyürek et al., 2023; Reddy, 2024), generalize differently from context than from weights (Chan et al., 2022a), use it to a degree set by training dynamics (Singh et al., 2023; Anand et al., 2024), and leave long contexts unread in the middle (Liu et al., 2024b; Hsieh et al., 2024). Whether a network uses its context is therefore a property produced by training, not one guaranteed by the architecture or by the presence of the information in the input. Outcome measures do not separate the two cases: a policy can reach the same success rate by reading its context or by predicting the same actions from the current observation alone, and only an intervention on the context tells them apart. Interventional diagnosis of this kind has exposed such shortcuts in vision (Geirhos et al., 2020; Darcet et al., 2024) and in robot policies (Wang et al., 2026). These findings describe when in-context use emerges and how to detect it; ROBIN turns them into a training and retention procedure for a robot policy, and the same counterfactual intervention is the measurement by which it verifies that the deployed policy reads its memory.

3 METHOD

At test time the episode history is condensed into a short context g, and the policy acts on the current view and on g together (Figure 2). The memory is still written into g by the existing memory system; what ROBIN changes is whether the policy reads it. The section follows the figure from left to right. We first formalize the memory-conditioned policy and the generation of its context (§3.1–§3.2, Figure 2a). ROBIN then has two components: **noise-matched training** (NMT) trains the policy on the context it will face at deployment (§3.3, Figure 2b), and **counterfactual read-out probing** (CRP) keeps only a policy that demonstrably reads it (§3.4, Figure 2c).

3.1 MEMORY-CONDITIONED POLICIES AND THE READ-OUT GAP

A memory-conditioned policy $\pi_\theta(a_t \mid o_t, g_t)$ maps the current observation o_t and a memory read-out g_t to a chunk of actions, and is trained by imitation on demonstrations $\mathcal{D} = \{(o_t, a_t, g_t^*)\}$,

$$\mathcal{L}(\theta) \;=\; \mathbb{E}_{(o,a,g^*)\sim\mathcal{D}}\big[-\log \pi_\theta(a \mid o, g^*)\big]. \tag{1}$$

We call g_t the policy's *context*, and, following Chan et al. (2022b), we call the policy *in-context* when its actions causally depend on g_t and *in-weights* when they do not. The only input that distinguishes a memory-conditioned policy from a memoryless one is g_t.

The context takes two forms, and both are used unchanged. In the *symbolic* form, g_t is a single line of text, a subgoal $g = (z, c)$ that pairs a template $z \in \mathcal{Z}$ with an image coordinate c, rendered as

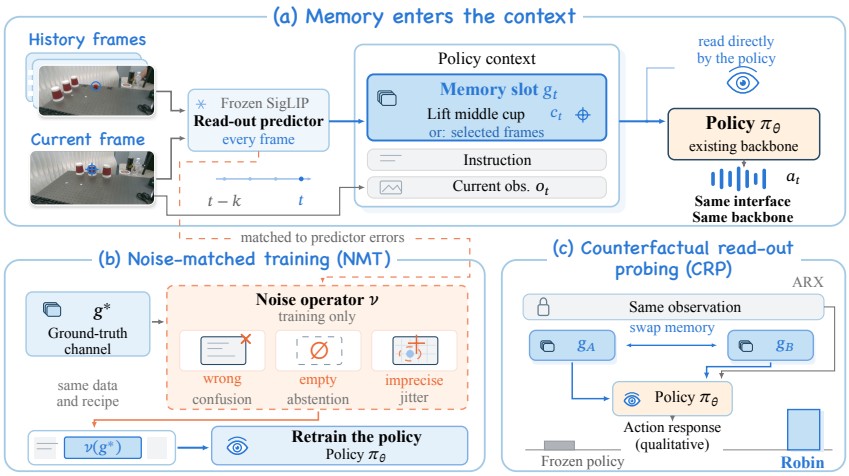

Figure 2: **ROBIN makes the policy read the memory it is given.** (a) Memory enters the context: the history is written as a one-line subgoal, or selected frames enter the image path, alongside the current view. (b) **Noise-matched training**: the policy is trained on that context in the form it will take at deployment, with the predictor's own error types and rates. (c) **Counterfactual read-out probing**: the policy is kept only if its actions change under the counterfactual probe, which swaps the memory with the view held fixed. Photographs: Franka; probe: ARX.

"pick up the container at $\langle y, x \rangle$*"* and appended to the language prompt. In the *frame* form, g_t is a small set of earlier frames chosen by a fixed rule and fed through the backbone's own image path next to the current view. Either way the memory does not bypass the policy: it is placed in front of it and the policy is free to use it or not.

The difficulty lies in where g_t comes from. During training the demonstrations provide a ground-truth context g_t^*, and this context is clean and stable: it names the right subgoal, it points at the right place, and it switches only when a subtask is completed, so within a subtask it is constant. At deployment no such signal exists. The context is produced frame by frame by a learned predictor, $\hat{g}_t$, which is correct most of the time but not always, and which may change its answer from one frame to the next. Writing $S_\theta(g)$ for the success of π_θ when its context is supplied by the process g, the quantity that matters is the read-out gap

$$\Delta(\theta) \;=\; S_\theta(g^*) \;-\; S_\theta(\hat{g}), \tag{2}$$

how much of what memory demonstrably offers is lost between the moment it is predicted and the moment the policy acts. Both terms are evaluated with the same policy, so Δ measures what better use of the same memory could recover, and it is what ROBIN sets out to reduce.

3.2 CONTEXT GENERATION: GROUNDED SUBGOAL PREDICTION AND FRAME SELECTION

The symbolic context is written by a predictor Φ that reads a pooled stream M_t of past frames together with the current frame F_t and emits two things: which subgoal applies and where in the image it is to be executed,

$$\hat{g}_t \;=\; (\hat{z}_t, \hat{c}_t) \;=\; \Phi(M_t, F_t), \qquad \hat{z}_t \in \mathcal{Z}, \;\; \hat{c}_t \in \Omega. \tag{3}$$

The first is a choice among a task-specific set of templates restricted by the environment and the instruction; the second is a location on the image plane obtained from a coarse grid cell and a continuous offset within it. The predictor is trained on labeled demonstration frames, with on-policy corrections gathered in the manner of DAgger (Ross et al., 2011), and its output is exactly the line of text the policy receives. For the frame context the same role is played by a selection rule $\mathcal{T}(t)$ that names which past frames enter the image path,

$$g_t \;=\; \{\, o_\tau : \tau \in \mathcal{T}(t) \,\}, \qquad \mathcal{T}(t) \;=\; \{0\} \,\cup\, \mathcal{U}(t) \,\cup\, \mathcal{N}(t), \tag{4}$$

the first frame of the episode as an anchor, a sparse uniform sweep $\mathcal{U}(t)$ of the history, and a dense window $\mathcal{N}(t)$ of the most recent frames, so that the policy always sees where it started, what happened last, and something of what happened in the middle.

What matters for the rest of the method is not how the predictor is built but how it fails, because those failures are what the policy will meet at deployment. A grounded predictor errs in three ways. It can be *confused*, naming a wrong subgoal and, with it, a wrong location. It can *abstain*, returning an empty context when it is not confident enough to commit. And it can be *imprecise*, naming the right subgoal but placing it a little off. These three modes partition its outputs,

$$p_{\text{conf}} \;=\; \Pr[\hat{z}_t \neq z_t^*], \qquad p_{\text{abst}} \;=\; \Pr[\hat{g}_t = \varnothing], \qquad p_{\text{loc}} \;=\; 1 - p_{\text{conf}} - p_{\text{abst}}, \tag{5}$$

and the three rates, together with the spread of the localization error, can be measured once on held-out data. The third rate is the residual, so the operator below needs no fourth branch. The predictor writes g by the rule above and is treated as fixed from here on; the training that follows changes only how the policy reads that line (Figure 2a).

3.3 Noise-Matched Training under the Deployment Context Distribution

A policy trained on the clean context g^* meets a different context at deployment, and does one of two things: it follows the context into its errors, or it looks past g and acts on the current view alone. ROBIN therefore gives the context its deployment-time form already during training, which we call *noise-matched training*. The policy is retrained with the same demonstrations, objective, architecture, and text format as before; the ground-truth context is passed through a noise operator ν before it reaches the policy, in the spirit of noise injection for imitation learning (Laskey et al., 2017), and ν reproduces the predictor's three failure modes in the proportions in which the predictor makes them:

$$\nu(g^*) \;=\; \begin{cases} (z', c') \sim \mathrm{Unif}(\mathcal{Z}\backslash\{z^*\}) \times \mathrm{Unif}(\Omega) & \text{w.p. } p_{\text{conf}} \\ \varnothing & \text{w.p. } p_{\text{abst}} \\ (z^*,\, c^* + \varepsilon) & \text{w.p. } p_{\text{loc}}, \end{cases} \tag{6}$$

where Ω is the image plane and ε is a spatial jitter drawn from a mixture of uniform boxes at a few scales, $\varepsilon \sim \sum_k w_k \mathcal{U}([-r_k, r_k]^2)$. The first branch is confusion, the second abstention, the third imprecision. The branch probabilities are the three rates of Eq. 5, measured on the deployed predictor, so the policy is trained on a context whose error types and rates are those it will meet at deployment. Training minimizes the same imitation loss as Eq. 1 with the corrupted context in place of the clean one,

$$\mathcal{L}_\nu(\theta) \;=\; \mathbb{E}_{(o,a,g^*)\sim\mathcal{D}}\, \mathbb{E}_{g\sim\nu(g^*)}\big[-\log \pi_\theta(a \mid o, g)\big], \tag{7}$$

The rates are read from the predictor rather than chosen, which removes the free parameter a fixed noise level would introduce.

Under ν the context is useful but not authoritative, and the policy learns to weigh the subgoal against the current view: to go where the context points when the scene is consistent with it, and to fall back on what it sees when the two disagree. A policy trained this way reads the g the predictor emits directly, and keeps reading it after retrieval becomes imperfect. Under the frame form of the context the same principle applies with the selection rule in place of the predictor: the policy is trained on frames chosen by the rule it will meet at deployment, rather than on frames chosen some other way. In both forms the policy is trained on the context as it will arrive, a line of text with the predictor's errors or a set of frames in the deployed layout, and what it learns in either case is to read that context (Figure 2b).

3.4 Counterfactual Read-Out Probing for Policy Retention

Training loss and success rate do not settle whether a policy reads its context. A policy that has learned to solve a task from the current view alone can score well while ignoring g_t entirely, and two policies with the same success rate can differ completely in whether memory played any part in it. Since the whole purpose of ROBIN is a policy that uses its memory, that property has to be checked directly.

The check is an intervention, the *counterfactual probe*. Hold the observation fixed and change only the context: hand the policy the correct subgoal g, then a different one g', and compare the actions it produces,

$$\rho(\theta) \;=\; \mathbb{E}_o\, \mathbb{E}_{g'\neq g} \big\| \hat{a}_\theta(o, g) - \hat{a}_\theta(o, g') \big\|_2^2, \tag{8}$$

Table 1: RMBench success rate (%; 50 official demonstrations and 100 rollouts per task; best **bold**, runner-up underlined). Baselines as reported by Chen et al. (2026) and Yang et al. (2026); MemoryVLA is EventVLA's QwenOFT re-implementation. ROBIN is trained on the union of the nine tasks for 80k steps and continued per task (§4.1). †: entry uses inference-time sampling.

	M(1) tasks					M(n) tasks				Means		
Method	Obs&Pick	Rearr.	PutBack	SwapB	SwapT	Battery	Ranking	Cover	Press	M(1)	M(n)	All
DP (Chi et al., 2023)	1	0	0	11	20	10	10	0	0	6.4	5.0	5.8
ACT (Zhao et al., 2023)	1	29	0	2	2	19	0	0	0	6.8	4.8	5.9
$\pi_{0.5}$ (Physical Intelligence, 2025)	9	13	11	24	15	16	6	0	0	14.4	5.5	10.4
X-VLA (Zheng et al., 2025)	9	13	18	16	3	26	1	2	0	11.8	7.3	9.8
MemoryVLA (Shi et al., 2025a)	2	53	81	76	9	33	53	69	0	44.2	38.8	41.7
Mem-0 (Chen et al., 2026)	4	89	90	67	14	28	18	68	0	52.8	28.5	42.0
EventVLA (Yang et al., 2026)	21	96	95	**96**	87	**35**	**81**	**97**	3	79.0	**54.0**	67.8
ROBIN (ours)	**38**	**98**	**97**	90†	**96**	28	72†	86	**8**	**83.8**	48.5	**68.1**

where $\hat{a}_\theta(o, g)$ is the action chunk the policy predicts. If ρ is large the actions move with the context and the policy is reading it; if ρ sits at the level a memoryless policy would produce, the memory is present but out of use. This counterfactual test costs a handful of forward passes, requires no rollouts, and can be run on hardware as easily as in simulation. *Counterfactual read-out probing* keeps the policy whose actions respond to its context, rather than the one with the lowest loss, and that is the policy deployed and reported. The same test is what the experiments use to confirm, on the real robot, that the deployed policy reads its memory (Figure 2c).

4 EXPERIMENTS

We report the main comparisons against published memory systems on RMBench and RoboMME (§4.2), isolate the effect of noise-matched training under controlled corruption (§4.3), and close on two physical arms, where we also verify directly that the deployed policy uses its context (§4.4).

4.1 BENCHMARKS, TRAINING CONFIGURATION, AND PROTOCOL

We evaluate on two memory benchmarks and two physical arms (Figure 4). **Nine-task RMBench setup.** Following Chen et al. (2026), each of the nine tasks provides 50 official demonstrations, and success is scored over 100 frozen-seed rollouts. We initialize from $\pi_{0.5}$ (Physical Intelligence, 2025), train one policy on the union of the nine tasks for 80k steps (batch 64), and continue it per task; the reported checkpoint of each task is chosen on a held-out 20-rollout panel by success rate and the read-out probe ρ (Eq. 8), then frozen for the 100-rollout report. Baseline numbers are quoted from Chen et al. (2026) and Yang et al. (2026). Ranking and Swap-Blocks use inference-time sampling.

Sixteen-task RoboMME setup. Following Dai et al. (2026), one shared policy is trained on all 16 tasks for 80k steps (batch 64) from $\pi_{0.5}$ and scored over 50 test episodes per task; seed-matched comparisons use the validation panel (16 tasks $\times$ 10 episodes). The policy is then aligned to the deployed frame rule by a short fine-tuning at the same batch size. Baseline numbers are quoted from Dai et al. (2026). **Two-arm real-robot setup.** The ARX and Franka tasks use shared pre-generated episode orders with 10 episodes per condition; their policies train for 10k steps at batch 64. Throughout, the policy conditioned on the ground-truth context g^* is a reference only.

The branch probabilities of Eq. 6 are read from the predictor's error histogram on held-out episodes, and the frame rule of Eq. 4 uses offsets of 15–300 steps (Appendices B and E). Success rate (%) is the metric throughout; significance uses the exact McNemar test where per-episode records are paired and Fisher's exact test on aggregate counts otherwise, and a wrist-camera leakage audit is reported in Appendix D. The predictor's corrective samples and the seed-matched ablation panel both come from the validation split; the frame-channel results of §4.2 never query the predictor. The study comprises 31 training runs on A100, A800, and H200 nodes, about 8,000 GPU-hours, and roughly 49,000 evaluation rollouts (Appendix G).

Table 2: RoboMME success rate (%; best **bold**, runner-up underlined). Baselines as reported by Dai et al. (2026); ROBIN scored on the 798 test episodes with records (50 per task). The greyed row is those same weights read under the deployed layout before the alignment step of §4.2. ROBIN matches or exceeds FrameSamp–Modul, the strongest published configuration overall, on 7 of the 16 tasks.

	Counting				Permanence				Reference				Imitation				
Method	BinFill	PickX	SwingX	StopCube	VidUmsk	BtnUmsk	VidUmskS	BtnUmskS	PickHL	VidRepick	VidPlcBtn	VidPlcOrd	MoveCube	InsPeg	PatLock	RouteStk	Avg
$\pi_{0.5}$ (memoryless)	30.0	42.9	35.6	6.7	20.4	22.2	18.7	6.7	11.3	0.4	31.1	25.8	26.0	1.6	2.9	4.7	17.9
$\pi_{0.5}$ w/ past actions	26.7	58.3	26.7	4.7	30.7	23.7	20.7	16.0	12.3	8.7	24.0	18.7	34.0	1.0	4.0	5.7	19.7
SAM2Act+ (Fang et al., 2025)	40.0	76.0	25.3	0.0	27.3	32.0	18.0	**26.7**	17.3	5.3	24.7	20.0	29.3	0.0	0.0	0.0	21.4
MemER (Sridhar et al., 2026)	**56.7**	79.3	59.3	0.0	81.3	**72.0**	**38.0**	21.3	**70.7**	25.3	30.0	26.0	**82.7**	6.7	16.7	12.0	42.4
GroundSG–QwenVL	52.0	92.7	7.3	0.0	**88.7**	24.0	30.7	14.0	15.1	25.3	54.0	31.8	71.6	3.3	6.7	6.0	32.7
TokenDrop–Modul	34.4	83.6	86.0	5.3	28.2	29.3	28.4	21.3	21.3	22.0	59.6	**36.0**	62.0	7.1	32.4	51.6	38.0
FrameSamp–Modul	39.6	87.3	92.0	**42.0**	32.7	25.1	24.4	18.2	22.9	**30.4**	**60.0**	32.0	77.8	**7.6**	**53.6**	**66.7**	**44.5**
ROBIN (ours)	40.0	**94.0**	**98.0**	32.0	28.0	26.0	22.0	20.0	32.0	18.0	56.0	**36.0**	62.0	6.0	38.0	64.6	42.0
Same weights before alignment (control)	36.0	32.0	42.0	22.0	22.0	28.0	20.0	12.0	16.0	0.0	22.0	34.0	30.0	0.0	42.0	34.0	24.5

Table 3: Robustness to channel corruption on RoboMME (success %, 160 episodes per row, identical seeds). The frozen policy is trained on the annotated context and ROBIN under ν; the context channel is the same for both. **Bold**: better under each condition.

Context channel at deployment	Frozen policy	ROBIN	Δ
Deployed predictor, no post-processing	27.5	27.5	0.0
Deployed predictor + temporal decoder	15.6	**16.3**	+0.7
One wrong subgoal held all episode, 75% of episodes	16.9	**28.8**	+11.9
One wrong subgoal held all episode, 100% of episodes	3.1	**15.0**	+11.9
Wrong subgoal redrawn every frame, 75% of episodes	16.2	**31.9**	+15.6
Uninformative (placeholder) channel	6.9	**18.1**	+11.2

4.2 MAIN RESULTS ON MEMORY BENCHMARKS

Comparison on RMBench. Under the official setting of 50 demonstrations and 100 rollouts per task, ROBIN's shared nine-task $\pi_{0.5}$ reaches the best published M(1) average, 83.8, at a nine-task mean of 68.1 (Table 1). Memoryless policies stay below 15 on average. The gains fall where the policy must read a single distant observation and act on it much later: Observe&Pick 38 against 21, Swap-T 96 against 87, and Press Button 8 against 0–3. Rearrangement and Put Back, in which the event is recent, were already high and move by one or two points (98 and 97). ROBIN is also runner-up on Swap-Blocks ($90^{\dagger}$) among the M(1) tasks and on Ranking ($72^{\dagger}$) and Cover (86) among the M(n) tasks, so the advantage is not confined to the tasks it leads.

Comparison on RoboMME. Against memoryless policies and published memory systems (Table 2), ROBIN reaches 42.0 average success with the same $\pi_{0.5}$ backbone as the benchmark's own mechanisms, matching or exceeding the strongest published configuration on 7 of 16 tasks. The memoryless $\pi_{0.5}$ averages 17.9, and appending its own past actions adds under 2 points. The gains concentrate where the policy must track history: on the Counting suite, whose tasks ask how many times an event has occurred, ROBIN averages 66.0 against 65.2 for the strongest published mechanism, the highest of any system in the table, and it sets the best number on PickXtimes (94.0) and SwingXtimes (98.0). The Permanence, Reference, and Imitation suites ask where an object was, which one it was, and how a demonstration went; there ROBIN matches the best published number on VideoPlaceOrder (36.0) and averages 42.6 on Imitation, second in the table.

What the policy reads is the frame rule it was trained on. On RoboMME memory arrives as selected frames, so the context the policy must read is their temporal layout, and ROBIN trains under the rule the policy is deployed with: the frame-channel counterpart of Eq. 7. With the weights held fixed, the same policy scores 42.2 under the layout it was trained with and 24.5 under the deployed one (paired McNemar $p{=}3.6{\times}10^{-22}$, 14 of 16 tasks lower); training it under the deployed layout returns it to 42.0 (paired $p{=}5.8{\times}10^{-22}$, 14 of 16 tasks higher), recovering 17.5 of the 17.7 points the layout costs. A second checkpoint behaves the same way, 40.8 under its own layout against 15.5 under another (paired $p{=}4.6{\times}10^{-41}$, Appendix E). The greyed row of Table 2 is that control per task (Figure 10); the two tasks that do not gain differ by two and four points, so the layout is what the policy learns to read. Validation and test scores agree within 4 points at every checkpoint, so the reported numbers do not depend on choosing the checkpoint on test data.

4.3 ABLATIONS UNDER CONTROLLED CHANNEL CORRUPTION

Each row of Table 3 changes only the context the policy receives at deployment; benchmark, tasks, and the 160 seeds are identical for both policies, so every Δ is a paired difference and the only

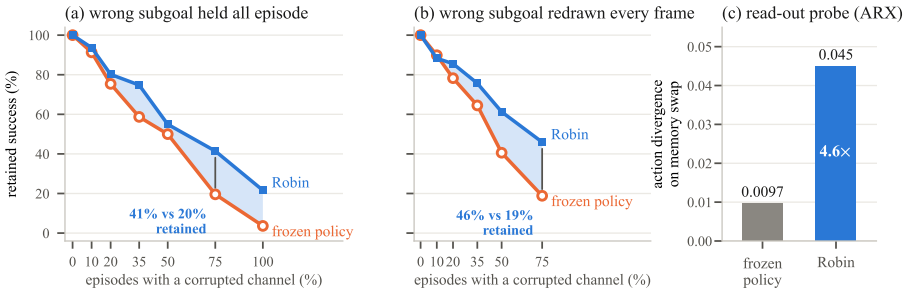

Figure 3: (a, b) Success retained under channel corruption on RoboMME, relative to each policy's clean-channel score (160 episodes per point); shading: margin retained by ROBIN. (c) Read-out probe on ARX: action divergence under a memory swap, frozen policy (0.0097, noise floor) vs. ROBIN (0.045).

variable is whether the policy was trained under ν. At the predictor's own operating point the two coincide by construction, both scoring 27.5 with no post-processing on either side; the rows below vary how unreliable that channel becomes, and there the two separate, at every level and by the widest margin at the most severe ones. With one wrong subgoal held for an entire episode in 75% of episodes, ROBIN scores 28.8 against the frozen policy's 16.9; at 100% it scores 15.0 against 3.1, almost five times as high, and in relative terms it retains 41% and 22% of its clean-channel success where the frozen policy retains 20% and 4%. Redrawing the wrong subgoal independently at every frame widens the margin further, to 31.9 against 16.2 (51/160 against 26/160, Fisher exact p=0.002), retaining 46% against 19%. The frozen policy is indifferent to the two regimes, scoring 16.9 when the error persists and 16.2 when it is resampled, whereas ROBIN gains three points from resampling, 28.8 to 31.9. A policy trained only on contexts that were always right has no alternative to following the subgoal, so its score tracks how often the subgoal is wrong and falls to near zero when it is always wrong, and a context that contradicts itself from frame to frame carries no signal it can use. Training under ν replaces following with weighing: the context becomes evidence to be checked against the current view, a subgoal that disagrees with the scene is discounted. The placeholder row isolates that mechanism. Given a context carrying no information at all, ROBIN scores 18.1, on a par with the memoryless 17.9, while the frozen policy falls to 6.9, well below what the same backbone reaches with no context at all (Δ=+11.2): a placeholder that looks authoritative is followed by the policy trained to follow and discounted by the one trained to weigh. In both panels of Figure 3, which plot the fraction of clean-channel success that survives, ROBIN's curve lies above the frozen policy's from 20% corruption onward: at the midpoint of the sweep the two retain 75% and 59% when the error is held and 76% and 64% when it is redrawn, and at the severe end 41% and 20%.

The alternative is to clean the channel before it reaches the policy, and the same table measures what that buys. A hysteresis decoder (Eq. 9) lowers the frozen policy by 12 points and lowers the retrained policy at every one of seven checkpoints (sign test, p=0.016); the retrained policy stays the better of the two under it (16.3 against 15.6), but both sit far below the 27.5 they reach when the predictor's output is read as it comes. The seven other configurations we ran fall below that level as well, the best reaching 15.6. All of them act on the symbol after it is emitted, and none gives the policy a way to trade it off against the view (Appendix A), so the weighing has to be learned during training rather than imposed after the context is written. The same holds on the frame channel, where ν is replaced by the deployed frame rule: the alignment of §4.2 is this procedure on a second context format, not a second method, and its control, those same weights read under the deployed layout before that training, at 24.5, is the frame-channel analogue of the rows above.

4.4 REAL-ROBOT EVALUATION

Two tasks make the current view uninformative by construction (Figure 4, Table 4; protocols in Appendix C).

Retrieval with the target out of view. Four snacks sit behind closed drawers on an ARX arm; the target is named only in a demonstration prefix and every drawer is closed at execution, so chance is



Figure 4: The two real-robot tasks end to end. Left of the dashed line the target is still in view; right of it the policy acts on a scene that no longer shows it. ARX: the target snack is shown while a drawer is open, then every drawer is closed. Franka: the cube is visible for two seconds, then covered at all three positions. Frames left of the line and the full Franka row are teleoperated recordings; the three ARX frames on the right are a ROBIN rollout (top-chips subclass). Full frames in Appendix C.

25% (protocol in Appendix C). The memoryless policy stays at chance (25.0); both ROBIN variants, subgoal line and frames, reach 47.5 (ground-truth class 72.5).

Table 4: Real-robot success (%, 10 episodes per cell; best deployable in **bold**; ground-truth rows get the answer from the operator). *Left:* ARX retrieval, chance 25%. *Right:* Franka cup-and-cube, first contact per position (L/M/R) and over all 30, chance 33%.

	LP	LU	TC	TN	Avg
$\pi_{0.5}$ (memoryless)	20	40	30	10	25.0
ROBIN, frame memory	50	50	60	30	**47.5**
ROBIN, subgoal channel	60	60	40	30	**47.5**
Ground-truth class	90	80	70	50	72.5

	L	M	R	First-contact	Uncover
$\pi_{0.5}$ (memoryless)	90	0	20	36.7	23.3
ROBIN	70	60	70	**66.7**	**53.3**
Ground-truth position	90	80	90	86.7	70.0

Occlusion with a symmetric scene. A cube is shown at one of three positions on a Franka arm and identical cups are then inverted over every position; first contact is scored before any grasp and chance is 33%. The memoryless policy is at chance (11/30). ROBIN contacts the correct cup in 20/30 episodes (Fisher exact p=0.038), evenly over the three positions (7, 6, 7), and uncovers the cube in 16/30 against 7/30 (p=0.033), against a ground-truth reference of 26/30 and 21/30.

Causal dependence on memory. Holding the observation fixed and swapping the memory moves ROBIN's action chunk 4.6 times as far as the frozen policy (0.045 against a noise floor of 0.0097; CRP, Figure 3c). Removing the demonstration prefix raises the action error by 38.6% over the first 30 steps, where the drawer is closed and the target has to be recalled, against 2.7% over the full episode, so the response is concentrated where memory is needed and whole-episode averages dilute it by an order of magnitude (Appendix C).

5 CONCLUSION

Memory already reaches the policy as the context in front of it; ROBIN trains the policy on that context in the form it takes at deployment and keeps the one whose actions depend on it. It reaches 42.0 on RoboMME and the best published M(1) average on RMBench, 83.8, and its actions move when the memory is swapped under a fixed view. Held at fixed weights, the same policy loses 17.7 points when the layout of its context changes and recovers them once it is trained under the new layout. On two arms whose view cannot decide the action, success rises from 25.0 to 47.5 in retrieval and first contact from 36.7 to 66.7 under occlusion. Benchmarks report what a policy stores; the probe tests whether it reads, and training it for the memory it will receive is what keeps memory in-context.

AI USE STATEMENT

In this work, we used generative AI tools only to edit author-written text for grammar, style, and readability. The authors reviewed all AI-assisted revisions and take full responsibility for the final manuscript, including its claims, citations, and experimental reporting.

ETHICS STATEMENT

This work studies memory read-out in robot policies on simulated benchmarks and two robot arms operated under standard laboratory safety procedures (operator presence, emergency stop, speed-limited motion). No human-subject data is involved. Improved long-horizon robot control has dual-use potential common to robot learning; we do not identify risks specific to this method beyond that baseline.

REPRODUCIBILITY STATEMENT

Section 4.1 describes the benchmark protocols, data splits, and evaluation settings. Appendices B, C, E, and G document data preparation, context construction, hyperparameters, training procedures, and compute requirements. The counterfactual read-out probe is defined in Section 3.4; corruption sweeps and real-robot diagnostics are reported in Sections 4.3 and 4.4. Appendix A specifies the post-processing configurations and thresholds, and Appendix D reports the wrist-camera leakage audit. Validation-set results and published baseline numbers are explicitly identified. Training and evaluation code is included in the supplementary material and will be publicly released.

REFERENCES

Michael Ahn, Anthony Brohan, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *CoRL*, 2022.

Ekin Akyürek, Dale Schuurmans, et al. What learning algorithm is in-context learning? investigations with linear models. In *ICLR*, 2023.

Suraj Anand, Michael A. Lepori, et al. Dual process learning: Controlling use of In-Context vs. In-Weights strategies with weight forgetting. *arXiv preprint arXiv:2406.00053*, 2024.

Samy Bengio, Oriol Vinyals, et al. Scheduled sampling for sequence prediction with recurrent neural networks. In *NeurIPS*, 2015.

Kevin Black, Noah Brown, et al. π_0: A Vision-Language-Action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024a.

Kevin Black, Mitsuhiko Nakamoto, et al. Zero-Shot robotic manipulation with pretrained Image-Editing diffusion models. In *ICLR*, 2024b.

Anthony Brohan, Noah Brown, Justice Carbajal, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *CoRL*, 2023a.

Anthony Brohan, Noah Brown, et al. RT-1: Robotics transformer for Real-World control at scale. In *RSS*, 2023b.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Learning to act anywhere with task-centric latent actions. In *RSS*, 2025.

Stephanie C. Y. Chan, Ishita Dasgupta, et al. Transformers generalize differently from information stored in context vs in weights. *arXiv preprint arXiv:2210.05675*, 2022a.

Stephanie C. Y. Chan, Adam Santoro, Andrew K. Lampinen, et al. Data distributional properties drive emergent in-context learning in transformers. In *NeurIPS*, 2022b.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Qiwei Liang, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

Tianxing Chen, Yuran Wang, Mingleyang Li, Yan Qin, Hao Shi, et al. Rmbench: Memory-dependent robotic manipulation benchmark with insights into policy design. *arXiv preprint arXiv:2603.01229*, 2026.

Egor Cherepanov, Nikita Kachaev, et al. Memory, benchmark & robots: A benchmark for solving complex tasks with reinforcement learning. *arXiv preprint arXiv:2502.10550*, 2025.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *RSS*, 2023.

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.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *ICLR*, 2024.

Maximilian Du, Suraj Nair, et al. Behavior retrieval: Few-Shot imitation learning by querying unlabeled datasets. In *RSS*, 2023a.

Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *Advances in Neural Information Processing Systems*, 2023b. arXiv:2302.00111.

Haoquan Fang, Markus Grotz, Wilbert Pumacay, Yi Ru Wang, Dieter Fox, Ranjay Krishna, and Jiafei Duan. Sam2act: Integrating visual foundation model with a memory architecture for robotic manipulation. In *ICML*, 2025.

Shivam Garg, Dimitris Tsipras, et al. What can transformers learn In-Context? a case study of simple function classes. In *NeurIPS*, 2022.

Robert Geirhos, Jörn-Henrik Jacobsen, et al. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2020.

Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. In *NeurIPS*, 2019.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint*, 2023. arXiv:2301.04104.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Cheng-Ping Hsieh, Simeng Sun, et al. RULER: What's the real context size of your Long-Context language models? In *COLM*, 2024.

Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations. *arXiv preprint*, 2024. arXiv:2412.14803.

Moo Jin Kim, Karl Pertsch, et al. OpenVLA: An Open-Source Vision-Language-Action model. In *CoRL*, 2024.

Michael Laskey, Jonathan Lee, et al. DART: Noise injection for robust imitation learning. In *CoRL*, 2017.

Fanqi Lin, Yingdong Hu, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation. In *ICLR*, 2025.

Fangchen Liu, Kuan Fang, et al. MOKA: Open-World robotic manipulation through Mark-Based visual prompting. In *RSS*, 2024a.

Nelson F. Liu, Kevin Lin, John Hewitt, et al. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 2024b.

Corey Lynch, Mohi Khansari, et al. Learning latent plans from play. In *CoRL*, 2019.

Steven Morad, Ryan Kortvelesy, et al. POPGym: Benchmarking partially observable reinforcement learning. In *ICLR*, 2023.

Soroush Nasiriany, Fei Xia, Wenhao Yu, et al. Pivot: Iterative visual prompting elicits actionable knowledge for vlms. *arXiv preprint arXiv:2402.07872*, 2024.

Tianwei Ni, Benjamin Eysenbach, and Ruslan Salakhutdinov. Recurrent Model-Free RL can be a strong baseline for many POMDPs. In *ICML*, 2022.

Tianwei Ni, Michel Ma, et al. When do transformers shine in RL? decoupling memory from credit assignment. In *NeurIPS*, 2023.

Octo Model Team. Octo: An Open-Source generalist robot policy. In *RSS*, 2024.

Catherine Olsson, Nelson Elhage, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

Physical Intelligence. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Marco Pleines, Matthias Pallasch, et al. Memory gym: Towards endless tasks to benchmark memory capabilities of agents. *arXiv preprint arXiv:2309.17207*, 2023.

Gautam Reddy. The mechanistic basis of data dependence and abrupt learning in an in-context classification task. In *ICLR*, 2024.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, et al. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2508.19236*, 2025a.

Lucy Xiaoyang Shi, Brian Ichter, et al. Hi robot: Open-Ended instruction following with hierarchical Vision-Language-Action models. *arXiv preprint arXiv:2502.19417*, 2025b.

Aaditya K. Singh, Stephanie C. Y. Chan, Ted Moskovitz, et al. The transient nature of emergent in-context learning in transformers. In *NeurIPS*, 2023.

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. Memer: Scaling up memory for robot control via experience retrieval. In *ICLR*, 2026.

Kai Wang, Zhaopeng Gu, Yixiang Chen, Yuan Xu, Qisen Ma, et al. Dim-wam: World-action modeling with diverse historical event memory. *arXiv preprint arXiv:2606.27677*, 2026.

Chuan Wen, Jierui Lin, et al. Fighting copycat agents in behavioral cloning from observation histories. In *NeurIPS*, 2020.

Ganlin Yang, Zhangzheng Tu, Yuqiang Yang, Sitong Mao, Junyi Dong, et al. Eventvla: Event-driven visual evidence memory for long-horizon vision-language-action policies. *arXiv preprint arXiv:2606.20092*, 2026.

Jianwei Yang, Hao Zhang, et al. Set-of-Mark prompting unleashes extraordinary visual grounding in GPT-4V. *arXiv preprint arXiv:2310.11441*, 2023.

Wentao Yuan, Jiafei Duan, et al. RoboPoint: A Vision-Language model for spatial affordance prediction for robotics. In *CoRL*, 2024.

Michał Zawalski, William Chen, et al. Robotic control via embodied Chain-of-Thought reasoning. In *CoRL*, 2024.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *RSS*, 2023.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. *arXiv preprint arXiv:2510.10274*, 2025.

A SERVE-TIME INTERVENTIONS ON THE PREDICTED CONTEXT

The post-processing interventions of §4.3 are combinations of the following mechanisms, applied to the frozen policy on the context emitted by the same predictor: a confidence gate on the template posterior; a hysteresis decoder that holds the predicted subgoal fixed until the predictor has been confident for several consecutive frames, switching the emitted state s_t only when

$$s_t = \begin{cases} \hat{z}_t & \text{if } \max_z p_t(z) \geq \tau \text{ for } K \text{ consecutive frames and } t - t_{\text{sw}} \geq C \\ s_{t-1} & \text{otherwise,} \end{cases} \tag{9}$$

with confidence threshold τ=0.85, confirmation window K, cooldown C, and t_{sw} the time of the last switch; coordinate locking between switches; a task-conditioned template mask; a prompt-compatibility mask; grammar-constrained decoding over a task-phase trie; and DAgger-style predictor retraining on 48k on-policy corrective samples. We sweep eight combinations of these mechanisms. Table 5 reports the two that are specified by the decoder of Eq. 9 alone, together with the reference score of reading the predicted context directly; the remaining six combinations score 5.6, 6.3, 8.1, 8.8, 12.5, and 12.6, so every intervention falls below that reference and the strongest of them is hysteresis decoding at 15.6. The per-combination, per-task matrix is listed with the code release. A confidence gate, a coordinate lock, and a decoder that waits for several agreeing frames all act on the symbol once the predictor has emitted it, so none of them gives the policy a way to weigh the symbol against the current view, and the waiting itself keeps a wrong subgoal in the context for longer.

Table 5: Post-processing interventions applied to the frozen policy on the predicted context (validation protocol, 16 tasks × 10 episodes). Success is the fraction of the episodes run. Every intervention scores below the 27.5 the same policy reaches when it reads the predicted context directly, and the strongest of them is hysteresis decoding. Which mechanisms each of the six remaining combinations applies is listed with the code release.

Intervention	Successes	Episodes run	Success (%)
None: predicted context read directly	44	160	**27.5**
Hysteresis decoding (Eq. 9), K=1, C=16	25	160	**15.6**
Hysteresis decoding (Eq. 9), K=2, C=48	9	141	6.4
Six further mechanism combinations	9–20	151–160	5.6–12.6

B SUBGOAL TEMPLATES AND PREDICTOR TRAINING

The template set $\mathcal{Z}$ contains 121 entries of the form "*verb* the *object* at $\langle y, x \rangle$". The template head is a masked posterior

$$p_t(z) \;=\; \frac{m_{\text{task}}(z)\, m_{\text{prompt}}(z)\, \exp f_z(M_t, F_t)}{\sum_{z'} m_{\text{task}}(z')\, m_{\text{prompt}}(z')\, \exp f_{z'}(M_t, F_t)}, \tag{10}$$

where M_t is the pooled memory stream, F_t the current frame, and $m_{\text{task}}, m_{\text{prompt}} \in \{0,1\}^{|\mathcal{Z}|}$ are hard masks derived from the environment identifier and the language prompt; they leave 4.9 candidates on average. Grounding factorizes into cell classification over a coarse grid and continuous offset regression,

$$\hat{c} \;=\; \text{center}(k^*) + \delta_{k^*}, \qquad k^* = \arg\max_k \; q_t(k), \tag{11}$$

with q_t the cell posterior and δ_k the offset. The predictor consumes precomputed SigLIP features only: a 4×4 pooled stream over the episode history for the template head and the current frame at

8×8 resolution for the grounding head. Both heads are trained with cross-entropy (template and cell) plus an L_1 loss on the continuous offset, on ground-truth-labeled frames augmented with 48k on-policy corrective samples collected DAgger-style on the validation split. The noise operator of Eq. 6 is applied to the ground-truth subgoal at policy-training time only; the predictor itself is never trained on corrupted targets.

Noise-operator settings. The operator of Eq. 6 uses branch probabilities $p_{\text{conf}}=0.08$ and $p_{\text{abst}}=0.05$ measured on held-out episodes, leaving $p_{\text{loc}}=0.87$ as the residual, with coordinate jitter $\varepsilon \sim 0.7\mathcal{U}[\pm8] + 0.2\mathcal{U}[\pm24] + 0.1\mathcal{U}[\pm64]$ pixels per axis. Checkpoints are chosen on a held-out panel by a 20-rollout scan and the probe of Eq. 8, then confirmed at 100 rollouts.

C REAL-ROBOT PROTOCOL

The probe also locates where memory is read: on held-out snack-retrieval episodes, removing the demonstration prefix raises the subgoal-channel policy's action error by 38.6% over the first 30 steps (drawer closed, target to be recalled) against 2.7% over the full 120-step episode (28.5% and 3.8% for a wrong prefix), so whole-episode averages dilute the read-out signal by an order of magnitude.

The ARX policy trains on 192 teleoperated episodes, of 204 recorded, over the four subclasses (left pink, left purple, top chips, top noodle snack); each episode carries a demonstration prefix of about 110 frames taken from a donor episode, and the donor, window, and prefix length are logged per episode. The Franka policy trains on 65 teleoperated episodes drawn from 150 recorded at 50 per cube position and averaging about 1,175 frames each, with nine episodes held out and one skipped; the training packing oversamples the decision window, resampling those 65 episodes to 184,176 frames, 2.4 times their native length. Both datasets are stored as precomputed features rather than raw frames.

ARX evaluation: four subclasses, 10 episodes each per arm, pre-generated balanced order shared across arms; episodes invalidated only for hardware faults. Franka: three positions, balanced blocks of three, all cups touched in a fixed order each episode, cups shuffled between episodes; first-contact and uncover scored separately. An ARX episode succeeds when the retrieved item belongs to the demonstrated subclass; a Franka episode scores a first contact when the first cup touched covers the cube, recorded before any grasp is attempted, and an uncover when the cube is exposed within the episode. Figures 5–7 show the scenes.

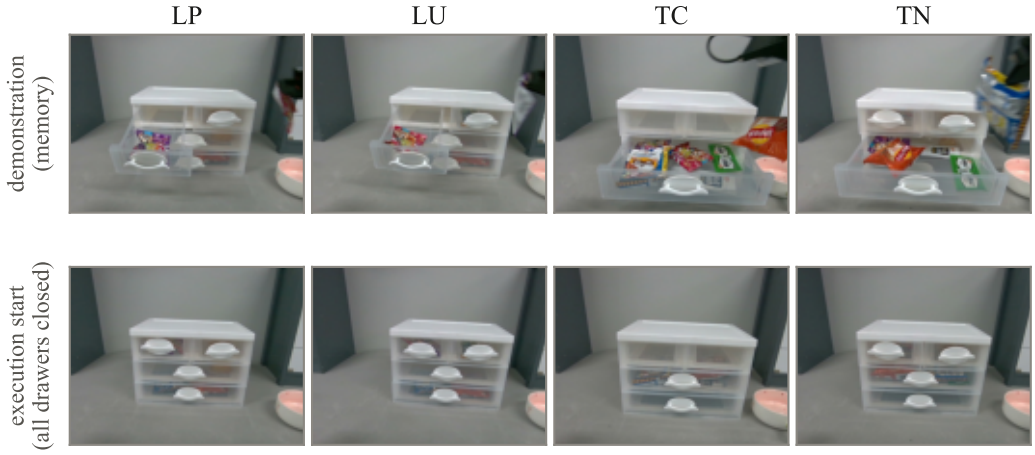

Figure 5: ARX four-way snack retrieval. Top: one demonstration frame per subclass with the drawer open and the target visible (LP left pink, LU left purple, TC top chips, TN top noodle snack). Bottom: the first frame of the same episode; all drawers are closed and the translucent fronts show only faint content. Fixed third-person camera, 640×480, cropped.

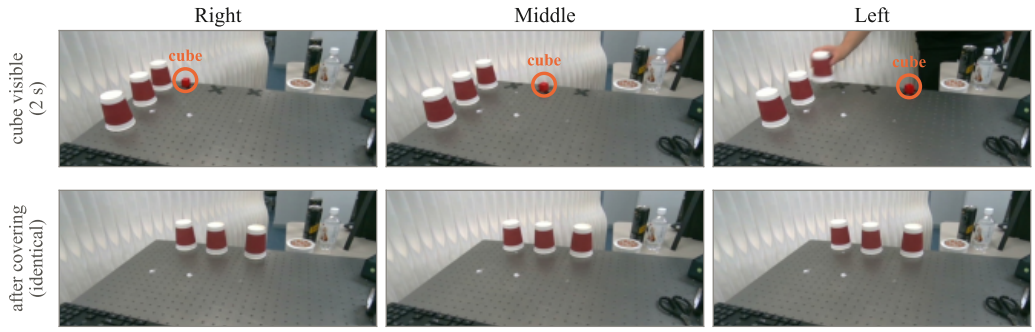

Figure 6: Franka cup-and-cube. Top: the cube (circled) at each of the three positions during the two-second observation window. Bottom: the same episodes after the operator has covered every position; the three scenes are visually identical. Frames from the fixed rear camera of teleoperated data-collection episodes; position names are given from the robot's side.



Figure 7: One Franka episode (middle position) from the fixed rear camera: covered scene, approach, first contact, and uncover with the cube revealed (circled). Teleoperated data-collection episode; policy rollouts are scored on first contact and on uncover (Table 4).

D WRIST-CAMERA LEAKAGE AUDIT

Zeroing both wrist-camera streams (with tensor shapes unchanged) on the same frozen seeds changes nothing: over ten paired runs we observe 128/200 successes with and without the intervention and zero discordant episodes, and at n=100 on Put Back, 95/100 both ways. With zero discordant pairs the McNemar test is undefined, so we report three-fold-rule upper bounds (1.5% over the pooled 200 episodes; 3.0% at n=100). Under the same intervention, the DiM-WAM audit (Wang et al., 2026) reports LingBot-VA falling from 86 to 41 on the same task. Our memory use therefore does not route through wrist-view shortcuts.

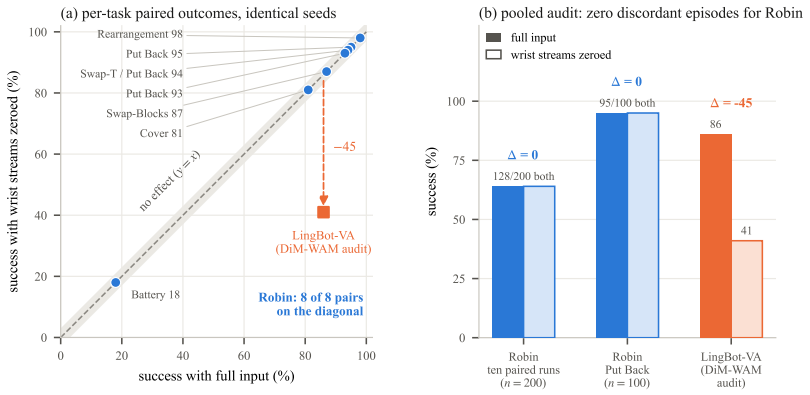

Figure 8: Wrist-camera leakage audit. (a) Success with both wrist streams zeroed (tensor shapes unchanged) against success with full input, per task on the same frozen seeds; every ROBIN pair lies on the diagonal, and the grey band marks ±3 points. The DiM-WAM audit (Wang et al., 2026) reports LingBot-VA falling from 86 to 41 under the same intervention. (b) Pooled counts: ten paired runs give 128/200 successes with and without the intervention, and Put Back at n=100 gives 95/100 both ways, i.e. zero discordant episodes.

E FRAME-SELECTION RULE

On RMBench the policy receives the episode's first frame (anchor) plus a dense-near window at offsets of 15, 30, 60, and 300 steps (0.3–6 s at 50 Hz). On RoboMME the deployed rule selects 32 frames including anchor and current, keeping 10 of 32 within the last 30 steps of a 600-step episode while bounding the largest gap at 26 steps, against 2 of 32 in the last 30 steps for even sampling.

The control of §4.2 reproduces on a second checkpoint: with the weights again held fixed, the 16-task mean is 40.8 under the layout that checkpoint was trained with and 15.5 under another (paired McNemar $p{=}4.6{\times}10^{-41}$, 15 of 16 tasks lower). Per-episode records for all five panels and the script that recomputes every number quoted here are included with the code release.

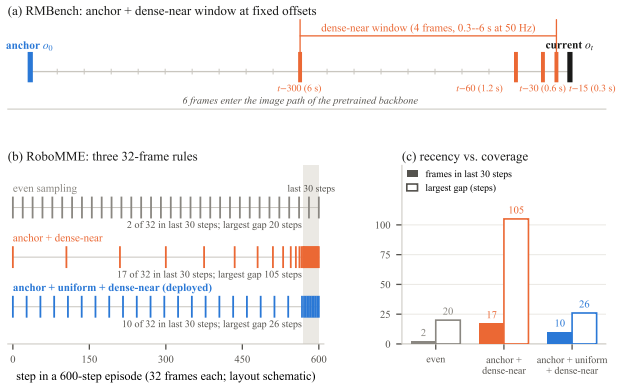

Figure 9: The frame-selection rule of §3.2. (a) RMBench: the policy receives the episode's first frame (anchor) and a dense-near window at offsets of 15, 30, 60 and 300 steps (0.3–6 s at 50 Hz) through the image path of the pretrained backbone. (b) RoboMME: all three rules select 32 frames including the anchor and the current frame; frame positions are schematic, the annotated counts are those of §4.2. Even sampling bounds the largest gap at 20 steps but places only 2 frames in the last 30 steps; anchor-plus-dense-near places 17 there but leaves gaps of 105 steps mid-episode; the deployed anchor-plus-uniform-plus-dense-near rule keeps 10 recent frames while bounding the largest gap at 26 steps. (c) The two coverage statistics side by side.

F ZERO-SHOT TRANSFER ON ROBOTWIN 2.0

This appendix reports the zero-shot performance on RoboTwin 2.0 of the same $\pi_{0.5}$ backbone used in the memory experiments. We train a single $\pi_{0.5}$ policy on 43 RoboTwin 2.0 tasks (Chen et al., 2025) (200 augmented demonstrations each) and evaluate it zero-shot on the seven held-out tasks of the benchmark, 100 episodes per task under the public seed protocol, with no task-specific data, prompts, or adaptation. Table 6 compares it with the published zero-shot systems WAN-Action, LingBot-VA, and Zero-WAM. Zero-WAM conditions on a human demonstration video of the target task at test time and reaches 46.95, so that comparison is not like for like. Without such a prompt, our policy averages 12.6 over the seven tasks, above WAN-Action's 10.98 and below LingBot-VA's 17.45. On Open Microwave it reaches 57.0, within two points of the video-prompted Zero-WAM (59.0) and more than twenty points above every other language-only system.

Table 6: Zero-shot success (%) on the seven held-out RoboTwin 2.0 tasks, 100 episodes per task under the public seed protocol. Baseline rows are mean ± std as reported by their authors; Zero-WAM receives a human demonstration video of the target task at test time, ours only the language instruction. Best per column in **bold**.

Method	Stapler→Pad	Open Microwave	Bread→Basket	Empty Cup	Object→Scale	Stack 3	Stamp Seal	Avg
WAN-Action	10.67±1.70	2.26±1.60	15.26±2.55	38.33±2.05	3.00±2.16	0.00±0.00	7.33±1.25	10.98±1.07
LingBot-VA	23.33±8.22	29.33±10.66	17.33±6.18	42.33±7.85	6.17±4.87	0.00±0.00	3.67±2.49	17.45±1.40
Zero-WAM	**69.14±2.93**	**59.00±2.83**	**35.00±3.74**	**84.87±0.18**	**24.67±2.05**	**9.00±2.16**	**47.00±4.55**	**46.95±0.70**
$\pi_{0.5}$ (ours, 43-task)	2.0	57.0	15.0	10.0	0.0	0.0	4.0	12.6

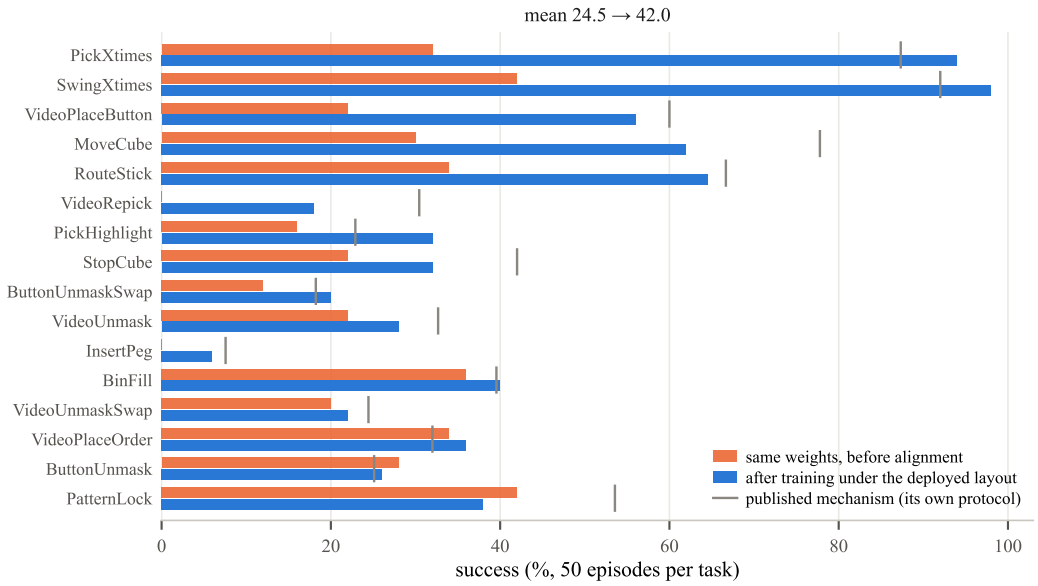

Figure 10: Per-task view of training under the deployed frame rule on RoboMME (Table 2), tasks sorted by gain. Orange: those same weights read under the deployed layout before the alignment step; blue: after training under that layout; grey ticks: the strongest published mechanism for each task under its own protocol. The blue bar is longer on 14 of the 16 tasks, and the 16-task mean is 42.0 against 24.5; VideoRepick has no orange bar because its value is 0.

G Training configuration and compute

Optimizer and schedule. All policies train with AdamW and gradient-norm clipping at 1.0 under a cosine schedule with 10,000 warmup steps, a peak learning rate of 5×10^{-5}, and 100,000 decay steps to a final rate equal to the peak, so the rate is constant after warmup over the step ranges used here. Weights are averaged with an exponential moving average (decay 0.99; 0.999 for the memoryless $\pi_{0.5}$ baseline). The action horizon is 20 and the batch size is 64 for the memory suite and 128 for the memoryless baseline. All training runs use seed 42; evaluation seed panels are fixed per task and held constant across the policies compared in any table.

Training and evaluation span 31 training runs on 8×A100-PCIE-40GB, 8×A800-SXM4-80GB, and H200 NVL nodes: about 7,000 A100-GPU-hours, 700 A800-GPU-hours, and 370 H200-GPU-hours, covering roughly 49,000 evaluation rollouts. The RoboMME policy follows the benchmark's published recipe (80k steps, batch 64; the first 55k steps on H200 nodes, continued to 80k on A100 at a measured 3.8 s/step), and aligning it to the deployed frame rule (§4.2) is a short fine-tuning at the same batch size, 14 hours against roughly 98 hours for the full recipe. RMBench uses a shared nine-task policy trained to 80k steps at batch 64 with per-task continuations (about 280k steps in total); per-task checkpoints are chosen by a 20-rollout scan and confirmed at 100 rollouts. Real-robot policies train at batch 64 and are evaluated at their 10k-step checkpoint; the RoboTwin 2.0 arm (Appendix F) trains for 30k steps (65 hours on A800).