DIVIDE-AND-REMEMBER: RECURSIVE ACTION-RELEVANT MEMORY FOR LONG-HORIZON VLA POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision–language–action (VLA) models struggle on history-dependent manipulation tasks, where the current observation alone does not determine the action, and the policy needs a memory of the history. Existing memory methods decide what to remember by design, for example, keeping frames with large pixel changes, and show inconsistent gains across tasks. We view what to remember as an optimisation problem. From the POMDP formulation of imitation learning, we show that the optimal memory maximises the conditional mutual information $I(a_t;\, m_t \mid o_t)$ between the action and the memory given the current observation. Intuitively, this means preserving the action-relevant information in the history that is not already contained in the current observation. Based on our analysis, we propose Divide-and-Remember (D&R), a recursive memory method that learns a memory function $m_t = \mathcal{M}(h_t)$ and scales to long contexts while staying compute-light. It involves two strategies: (1) the selection over the full history is divided recursively into subproblems of top-K selection over $2K$ tokens, so that fixed-size, lightweight selectors learned end-to-end support an unbounded history; (2) all recursion blocks share one selector, which captures the selection rule common to every block and keeps the method efficient. On RoboMME, a benchmark of 16 long-horizon manipulation tasks that require remembering when, where, what, and how to act, D&R achieves a state-of-the-art average success rate with consistent gains across all four suites under a budget of only 64 tokens; real-robot experiments show the same gain.

1 INTRODUCTION

Vision–language–action (VLA) policies are visuomotor policies trained by imitation learning, a.k.a. learning from demonstrations (LfD): a policy is learned from a dataset $\mathcal{D}$ of demonstrations, each an observation–action sequence $\{(o_t, a_t)\}_{t=1}^{T}$. Powered by large-scale robotic datasets and pretrained vision–language models, VLAs have achieved remarkable progress in robotic manipulation. However, many long-horizon tasks are partially observable, where the current observation o_t alone cannot determine the correct action. In Figure 1(a), the robot must move a cube "in the same manner as before", as shown in a video it has just watched; after either video it faces the *same* observation, and only the history tells it how to act. When the task's memory length, the shortest history the task requires, exceeds the policy's context length (Definition 1), such ambiguity can only be resolved by the history $h_t = \{(o_k, a_k)\}_{k<t}$. In this case, the policy needs **memory**, the ability to retain and reuse relevant information across time; we refer to this as the **memory problem**.

A growing body of research has integrated memory into VLAs, differing in what the memory encodes, from latent memory (Koo et al., 2026; Shi et al., 2026; Li et al., 2026) to perceptual memory (Yang et al., 2026; Dai et al., 2026), and alleviates the memory problem to some extent. However, what information the memory should carry remains fragmented. Latent memory learns a dense compression of the history h_t, which tracks temporal facts well (e.g., a count of events) but aligns poorly with the pretrained token space of the VLA and loses spatial detail. Perceptual memory keeps a subset of the history chosen by an inductive bias (e.g., the frames with the largest pixel change, or those picked by a VLM verifier); its verbatim snapshots preserve spatial facts (object locations and spatial relations) and object identity, but often miss some key robot behaviour, and the selection

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.

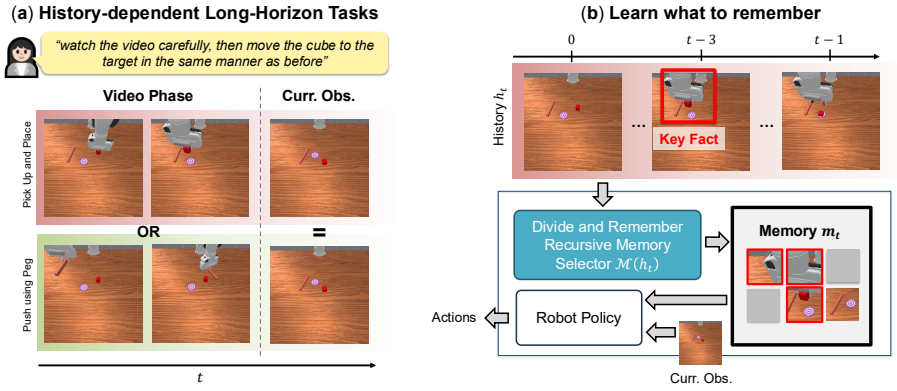


Figure 1: **Memory is a question of what to remember.** (a) A history-dependent task: "watch the video carefully, then move the cube to the target in the same manner as before". Two videos show different manners (pick-and-place versus peg push), yet at time step t the robot sees the *same* observation. A memoryless policy cannot recall how the cube was moved. (b) We view memory as an optimisation problem: find the memory $m_t = \mathcal{M}(h_t)$, selected from the full history, that carries the key fact the current observation lacks (here, the grasp pose at $t-3$), so that the policy knows how to move the cube.

is not optimised for control. This diversity reflects that existing work designs memory from task-specific (Yang et al., 2026; Zeng et al., 2026) perspectives, yielding inconsistent gains and a limited understanding of what the optimal formulation of memory is.

Starting from information theory and POMDPs, we cast the memory problem of a visuomotor policy as an optimisation problem: **the optimal memory maximises the conditional mutual information $I(a_t; m_t \mid o_t)$ between the action a_t and the memory m_t given the current observation o_t** (formal definition in Section 4); intuitively, it keeps the key facts that o_t lacks but a_t depends on, such as the grasp pose in Figure 1(b). This objective gives a unified view of existing memories. Perceptual heuristics keep the frames that differ most in pixels. This is a proxy for the conditional entropy $H(o_k \mid o_t)$, i.e., how much a candidate frame o_k differs from the current observation o_t. It is only a loose upper bound on what the frame adds, $I(a_t; o_k \mid o_t) \leq H(o_k \mid o_t)$, and spends the memory budget on information that does not help control, such as when the observation changes for reasons unrelated to the action (e.g., a lighting change). Latent memory does optimise $I(a_t; m_t \mid o_t)$ through its variational lower bound; the action log-likelihood $\mathbb{E}_{\mathcal{D}}[\log \pi_\theta(a_t \mid o_t, m_t)]$ is maximised jointly over the policy weights θ and the memory m_t, but it learns m_t in a new latent space and loses the spatial detail that the pretrained tokens of the VLA carry. Thus, we ask: *Can a memory be trained to maximise $I(a_t; m_t \mid o_t)$ directly, without heuristics, while staying in the pretrained token space of the VLA, so that it is optimised for control and still keeps the spatial details?*

We answer this question with **Divide-and-Remember (D&R)**, which optimises the memory against $I(a_t; m_t \mid o_t)$ in the pretrained token space of the VLA, so that choosing m_t becomes a discrete optimisation over the history tokens $\mathcal{H}_t$. D&R is a recursive memory method, in which the full history is divided recursively into slices of $2K$ tokens and a fixed-size, lightweight selector keeps K tokens of each slice. The K tokens kept from two neighbouring slices are merged into a new $2K$-token slice and selected again (see Figure 3), and the composition of these selections solves the memory selection over the full history. This design lets the history scale up while the learnable parameters stay light. The selector is trained end-to-end by the policy's action loss, the variational lower bound of $I(a_t; m_t \mid o_t)$.

We evaluate on RoboMME (Dai et al., 2026), a benchmark of 16 long-horizon manipulation tasks that each require a different fact from the past. Under a memory budget of only 64 tokens, D&R reaches a state-of-the-art average success rate of 38.6%, against 32.2% for the best latent memory and 17.9% for the memoryless policy. D&R leads on every memory demand of the benchmark, whether the policy has to accumulate a count over repeated events, track an object after it is hidden, refer back to an earlier moment of the episode, or reproduce a demonstrated sequence. We also

evaluate D&R on a real robot over several long-run tasks, with a person intervening mid-task and under perception noise, where D&R achieved the highest success rate.

2 Related work

Memory-augmented VLAs. Existing memory methods for VLAs differ in what the memory encodes, and each fixes what to remember by design. (1) *Latent memory* compresses the history into a compact representation, by merging each step into learnable moment tokens (HAMLET (Koo et al., 2026)) or by a recurrent belief state trained with world-model objectives (RB-VLA (Bagaria et al., 2026)); it tracks temporal facts well but loses spatial detail. (2) *Perceptual memory* keeps a subset of the raw history chosen by a fixed selection rule: frames a VLM labels as events (EventVLA (Yang et al., 2026)), the patches with the largest pixel change (TokenDrop (Dai et al., 2026)), or uniformly sampled frames (FrameSamp (Dai et al., 2026)); the snapshots preserve spatial and object detail, but the rule is hand-designed and not optimised for control. (3) *Parametric memory* folds the history into fast weights by test-time training (TTT) (Sun et al., 2025; Dai et al., 2026); a fact stored in weights is written by gradient descent, which is slow, and is hard to read back exactly.

Routes to the optimal POMDP policy. Classical solutions differ in which statistic of the history the policy conditions on. (1) The full history, $\pi(\cdot \mid o_t, h_t)$, by solving the associated dynamic decision network (Figure 2(a)); the policy grows exponentially with the history's length, so this works only for short histories (Poole & Mackworth, 2023). (2) The belief state, the posterior over the current state and a sufficient statistic for prediction, as in RB-VLA (Bagaria et al., 2026); its recurrent update makes compute and activation memory grow linearly with the episode length and cannot be parallelised over time, which is costly when every step runs a full forward pass of the VLA.

Our analysis gives all of these designs a common objective, $I(a_t;\, m_t \mid o_t)$; D&R is an approximation that optimises it in the pretrained token space, conditioning the policy on a bounded statistic, so that the memory stays small and the facts it keeps can be read back exactly.

3 Preliminaries

POMDPs and visuomotor policies. Instead of the state being observable, an agent gets only partial and/or noisy observations of the state before it has to act. Such a dynamic system is defined by a POMDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \Omega \rangle$ with states $s_t \in \mathcal{S}$, actions $a_t \in \mathcal{A}$, and observations $o_t \in \mathcal{O}$, as depicted by the decision diagram in Figure 2(a). The initial state is drawn from $p(s_0)$, the transition function $\mathcal{T}(s_{t+1} \mid s_t, a_t) = p(s_{t+1} \mid s_t, a_t)$ specifies the dynamics, and the observation function $\Omega(o_t \mid s_t) = p(o_t \mid s_t)$ specifies how observations arise from the state; all three are unknown.

A memoryless visuomotor policy $\pi_\theta(a_t \mid o_t)$, parameterised by θ, allows an agent to choose actions a_t in response to observations o_t to control a dynamic system, such as a robot. In LfD, we do not require any explicit reward function; we assume instead that the demonstrations come from an expert policy (Levine et al., 2016). The dynamics and the expert policy together induce a distribution over trajectories, each consisting of T observation–action pairs over a finite horizon, $\tau = \{(o_t, a_t)\}_{t=1}^{T}$. A demonstration dataset has N trajectories, $\mathcal{D} = \{\tau_i\}_{i=1}^{N}$. The training objective is to maximise the action log-likelihood under the expert data distribution:

$$\theta^\star = \arg\max_\theta \; \mathbb{E}_{(o_t,a_t)\sim\mathcal{D}}\big[\log \pi_\theta(a_t \mid o_t)\big]. \tag{1}$$

The policy comes from some parametric class parameterised by θ, which could be, for example, the weights of a VLA policy, $\pi_\theta(a_t \mid o_t, c)$, which maps a visual observation o_t and a language instruction c at time step t to a probability distribution over robot actions a_t.

Memory problems. Long-horizon robotic manipulation requires robots to compose perception, reasoning, and action over extended task sequences; however, not all of them need memory, the ability to retain and reuse historical context across time (Ni et al., 2023).

Definition 1 (memory problem for a visuomotor policy)**.** Write $h_{t-k:t} = \{(o_j, a_j)\}_{t-k\le j<t}$ for the last k steps of the history. The *memory length* l_{mem} of a task is the shortest window from which the expert policy can be realised,

$$l_{\mathrm{mem}} \;=\; \min\, k \quad \text{s.t.} \quad \pi^\star\big(a_t \mid h_t,\, o_t\big) = \pi^\star\big(a_t \mid h_{t-k:t},\, o_t\big) \;\; \forall t. \tag{2}$$

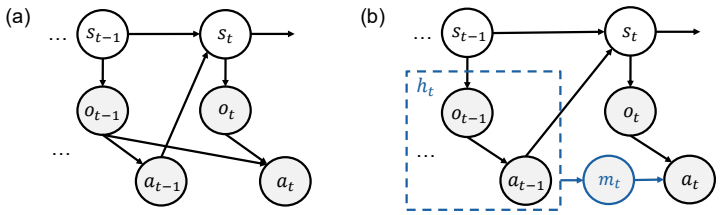


Figure 2: Graphical models of (a) imitation learning in a POMDP, where a_t depends on the history h_t, and (b) a memory-augmented policy, where m_t makes a_t independent of h_t given (o_t, m_t).

A policy that conditions on the last l steps, $\pi_\theta(a_t \mid h_{t-l:t}, o_t)$, has *context length* $l_{\text{ctx}} = l$ (Ni et al., 2023). A policy faces a **memory problem** if and only if the task's memory length exceeds the policy's context length, $l_{\text{mem}} > l_{\text{ctx}}$, i.e., the input is insufficient to determine the correct action, and the policy must retain and reuse historical context to resolve the ambiguity.

4 Optimal memory analysis

This section derives the objective behind "what to remember": the information a memory $m_t = \mathcal{M}(h_t)$ must carry so that a_t is independent of h_t given (o_t, m_t) (Figure 2(b)):

Proposition 1 (optimal memory for visuomotor policies)**.** *The information that the history h_t carries for control beyond the current observation is the conditional mutual information $I(a_t; h_t \mid o_t) = \mathbb{E}_{p(h_t,o_t)}[D_{\text{KL}}(\pi^\star(\cdot \mid o_t, h_t) \,\|\, p(\cdot \mid o_t))]$, the expected divergence between the expert policy and the action distribution obtained by marginalising it over the history. Let the memory be any function of the full history, $m_t = \mathcal{M}(h_t)$, inserted into the graphical model of Figure 2(b). By the chain rule, the memory splits this information into the part it keeps and the part it discards,*

$$I(a_t;\, h_t \mid o_t) \;=\; I(a_t;\, m_t \mid o_t) \;+\; I(a_t;\, h_t \mid o_t, m_t). \tag{3}$$

*Since the left-hand side does not depend on the memory, maximising $I(a_t;\, m_t \mid o_t)$ is equivalent to minimising the discarded information $I(a_t;\, h_t \mid o_t, m_t)$, so the **optimal memory for visuomotor policies** is*

$$\mathcal{M}^\star = \arg\max_{\mathcal{M}} \; I(a_t;\, \mathcal{M}(h_t) \mid o_t). \tag{4}$$

The memory $m_t = \mathcal{M}^\star(h_t)$ is a policy-sufficient statistic, i.e., $\pi^\star(a_t \mid o_t, h_t) = \pi^\star(a_t \mid o_t, m_t)$ for all (o_t, h_t).

Other solutions to POMDPs assume richer statistics of the history, for instance a belief that also predicts reward and transition; the more a statistic must carry, the more it demands of the model, the compute, and the memory budget (Appendix A).

The objective gives a unified view of existing memories. Perceptual memory keeps the frames that differ most from the current one, which maximises $H(o_k \mid o_t)$, an upper bound on what a past frame o_k adds since $I(a_t;\, o_k \mid o_t) \le H(o_k \mid o_t)$. The bound is loose when the observation changes for reasons unrelated to the action (e.g., a lighting change) (Dunion et al., 2023) and when the robot's behaviour is barely visible (e.g., a button press).

To see what can actually be optimised, we unfold $I(a_t;\, m_t \mid o_t)$:

$$\begin{aligned} I(a_t;\, m_t \mid o_t) &= H(a_t \mid o_t) \;-\; H(a_t \mid o_t, m_t) && \text{(definition)} && (5)\\ &= \mathbb{E}_{\mathcal{D}}\big[\log p(a_t \mid o_t, m_t)\big] + H(a_t \mid o_t) && \text{(expand the residual)} && (6)\\ &\ge\; \mathbb{E}_{\mathcal{D}}\big[\log \pi_\theta(a_t \mid o_t, m_t)\big] + H(a_t \mid o_t) && \text{(variational } \pi_\theta \text{ for } p) && (7) \end{aligned}$$

Equation 5 and Equation 6 are defined under the true $p(a_t \mid o_t, m_t)$ of the demonstration distribution, which is unknown and cannot be evaluated directly. Equation 7 replaces p with the policy π_θ, giving a lower bound that tightens as π_θ fits the data, while the entropy $H(a_t \mid o_t)$ is fixed by the data. Optimising a mutual-information objective through such a tractable bound has proven effective

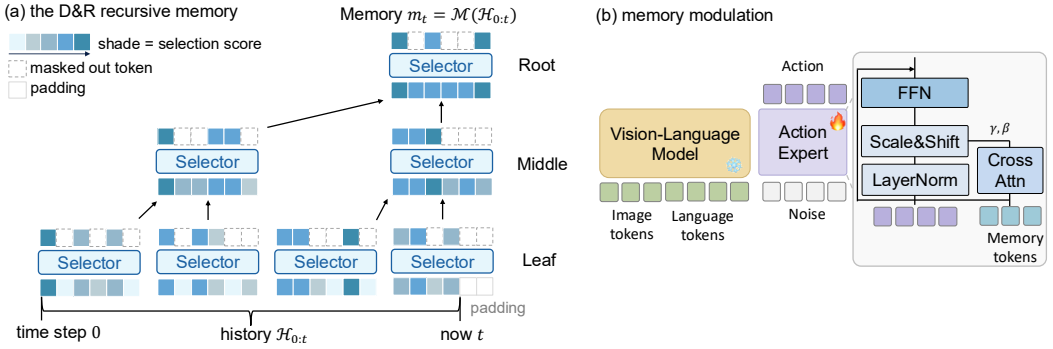


Figure 3: Overview of D&R. (a) At time step t, the selection over the full history $\mathcal{H}_t$ is divided recursively into subproblems of top-K selection over $2K$ tokens, each solved by the same lightweight selector f_ψ; the kept tokens are merged pairwise and re-selected level by level, and at the root the selected tokens are passed to the policy through the attention mask. Tokens keep their temporal order from left to right at every level. (b) D&R and all baselines modulate the memory into the action expert through an adaptive LayerNorm, where action features cross-attend to the memory tokens.

for generalisable control in reinforcement learning (Yu et al., 2024). **The objective becomes maximising $\mathbb{E}_{\mathcal{D}}[\log \pi_\theta(a_t \mid o_t, m_t)]$ over *two* variables: the policy parameters θ and the memory m_t**:

$$\max_{\mathcal{M}} I\big(a_t;\, m_t \mid o_t\big) \;\geq\; \max_{\mathcal{M},\,\theta} \mathbb{E}_{\mathcal{D}}\big[\log \pi_\theta(a_t \mid o_t, m_t)\big] + H(a_t \mid o_t), \tag{8}$$

where the right-hand side is the negative action loss and a constant. In practice $\pi_{0.5}$ (Black et al., 2025) trains with a flow-matching loss, which serves as a surrogate for the action log-likelihood here (it trains the same conditional action distribution but does not bound its likelihood). Optimising the right-hand side of Equation 8 through that loss opens a design space in which policy and memory are trained together end-to-end, *without* any auxiliary loss or hand-designed rule. Latent memory trained with the action loss, as HAMLET is, already lives in this space, but it optimises the bound through a fixed compression of every step, and so loses the fidelity of the pretrained tokens.

5 Divide-and-Remember

This section describes how we optimise $I(a_t;\, m_t \mid o_t)$ in practice. We keep the memory in the pretrained token space of the VLA, so that m_t is a subset of the history tokens and a kept fact can be read back exactly; memory optimisation then becomes a discrete problem, learning the mapping that selects, from the history tokens, the subset that maximises $I(a_t;\, m_t \mid o_t)$ under a budget of K tokens. Realising such an $\mathcal{M}$ raises two challenges: (i) $m_t = \mathcal{M}(h_t)$ is a function of the whole h_t, which grows without bound, yet $\mathcal{M}$ must stay *lightweight*, and (ii) $\mathcal{M}$ must bridge an *offline–online mismatch*: during training the whole history of an episode is available at once from $\mathcal{D}$, whereas at deployment it arrives one step at a time. Section 5.1 casts memory as budgeted token selection and introduces D&R that meets both challenges; Section 5.2 describes the selector and its training.

5.1 Memory as Budgeted Token Selection

Let $\mathcal{H}_t$ denote the set of all history tokens available at time step t. Index these tokens in temporal order, and let $\mathcal{H}_{i:j} \subseteq \mathcal{H}_t$ denote the tokens at positions i through j. A memory is a mapping $\mathcal{M}$ from the history tokens to a subset of at most K of them, $m_t = \mathcal{M}(\mathcal{H}_t) \subseteq \mathcal{H}_t$, $|m_t| \leq K$; let $\mathfrak{M}_K$ denote the set of such mappings. Writing $I(\mathcal{M})$ for $I(a_t;\, \mathcal{M}(\mathcal{H}_t) \mid o_t)$, the memory problem is

$$\mathcal{M}^\star = \arg\max_{\mathcal{M} \in \mathfrak{M}_K} I\big(a_t;\, \mathcal{M}(\mathcal{H}_t) \mid o_t\big). \tag{9}$$

The difficulty is that I is not available in closed form, so we settle for an approximate solution: our approach uses a parametric family $\mathcal{M}_\psi$ built from a selector learned end-to-end, optimised over ψ. Moreover, $\mathcal{H}_t$ grows with the history while a selector has a fixed input size. We fix the selector's input at $2K$ tokens and build $\mathcal{M}_\psi$ recursively: the selection over the full history is composed from

Algorithm 1 Divide-and-Remember

Require: span $\mathcal{H}_{i:j}$; budget K; selector f_ψ
Ensure: $\mathcal{M}_\psi(\mathcal{H}_{i:j}) \subseteq \mathcal{H}_{i:j}$, at most K tokens
if $|\mathcal{H}_{i:j}| \le K$ **then**
  return $\mathcal{H}_{i:j}$
end if
$\ell \leftarrow \max\{\, \ell' \ge 0 \;:\; 2^{\ell'} K < |\mathcal{H}_{i:j}| \,\}$
$q \leftarrow i + 2^\ell K - 1$ ▷ left-aligned split
return $f_\psi\big(\mathcal{M}_\psi(\mathcal{H}_{i:q}) \cup \mathcal{M}_\psi(\mathcal{H}_{q+1:j})\big)$

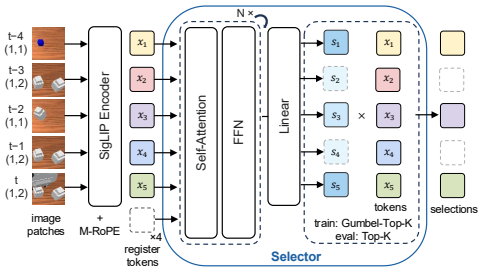


Figure 4: The selector f_ψ.

subproblems of top-K selection over $2K$ tokens, so one fixed-size, lightweight selector serves an unbounded history.

We instantiate this approach in a method we call **Divide-and-Remember** (D&R). Writing f_ψ for the learned selector that maps $2K$ tokens to their top-K (Section 5.2), the memory of a span $\mathcal{H}_{i:j}$ is

$$\mathcal{M}_\psi(\mathcal{H}_{i:j}) = \begin{cases} \mathcal{H}_{i:j}, & \text{if } |\mathcal{H}_{i:j}| \le K \\ f_\psi\big(\mathcal{M}_\psi(\mathcal{H}_{i:q}) \cup \mathcal{M}_\psi(\mathcal{H}_{q+1:j})\big), & \text{otherwise} \end{cases} \tag{10}$$

where the split is left-aligned, $q = i + 2^\ell K - 1$ with $2^\ell K$ the largest block size that fits strictly inside $\mathcal{H}_{i:j}$, and the memory at step t is $m_t = \mathcal{M}_\psi(\mathcal{H}_t)$. Because $\mathcal{M}_\psi$ composes over token spans rather than over a fixed window, training and deployment build the same D&R tree for the same prefix $\mathcal{H}_t$ of an episode, so $\mathcal{M}_\psi$ is free of the offline–online mismatch. Figure 3(a) illustrates the construction and Algorithm 1 gives the pseudocode.

The recursion filters tokens hierarchically from the leaves to the root: only tokens kept at one level are re-scored at the next, so computation is spent only on the tokens that survive. Higher levels thus hold increasingly important and sparse information, while the least important, such as background, is dropped at the leaves.

5.2 Learning the Selector

The selector f_ψ takes $2K$ candidate history tokens $\{x_i\}_{i=1}^{2K}$ in temporal order, encoded by the frozen SigLIP encoder, and returns the top-K of them as the memory of that slice. To preserve the spatiotemporal structure of the tokens, we adopt multimodal rotary position embedding (M-RoPE) (Wang et al., 2024) over time, height, and width. A transformer of N self-attention and FFN blocks then scores each token, and a linear head outputs a selection score s_i. The memory is

$$\hat{m} = \{\, g_i\, x_i \mid i \in \mathrm{TopK}(\{s_j\}_{j=1}^{2K}, K) \,\}, \tag{11}$$

$$g_i = \mathrm{sg}\left(g_i^{\text{hard}} - g_i^{\text{soft}}\right) + g_i^{\text{soft}}, \tag{12}$$

where $g_i^{\text{hard}} = \mathbb{1}[s_i > \rho]$ is the hard selection, $g_i^{\text{soft}} = \sigma((s_i - \rho)/\tau)$ its sigmoid relaxation with temperature τ, sg stops the gradient, and ρ is the threshold, the detached midpoint between the K-th and $(K{+}1)$-th largest scores. Selected tokens thus have $g_i = 1$ in the forward pass and receive gradient through g_i^{soft} in the backward pass (Rao et al., 2021). During training, Gumbel noise is added to s_i before the top-K for exploration (Kool et al., 2019); at the root, g_i masks the attention over the $2K$ candidates, and the masked candidates $\hat{m}$ are passed to the policy. A single selector f_ψ is shared across all nodes to keep training and inference efficient, and its only training signal is the flow-matching action loss of the policy that reads the selected tokens, the surrogate of the variational lower bound of $I(a_t;\, m_t \mid o_t)$ in Equation 8.

6 Experiments

Our experiments aim to answer whether D&R improves consistently across tasks with different memory demands, and where each memory method helps and why.

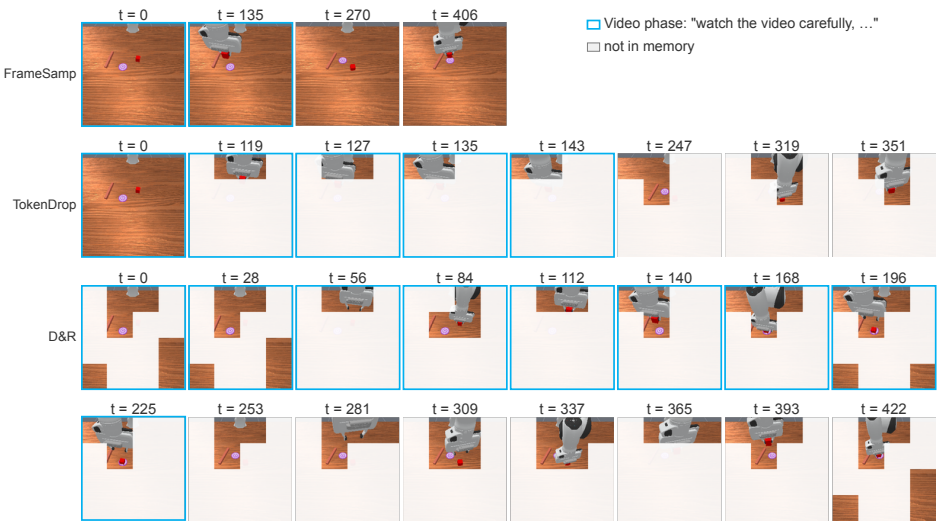


Figure 5: Memory at step 422 of a MoveCube episode under a 64-token budget, with the task goal "watch the video carefully, then move the cube to the target in the same manner as before". FrameSamp keeps 4 evenly spaced frames. TokenDrop keeps the first frame and spends the rest of the budget on the patches whose pixels changed most. D&R selects 64 patches recursively, 35 of them from the video phase. Kept patches are shown, white ones are not in memory, and light blue borders mark the video phase. All tasks are in Appendix B.3.3.

6.1 Experimental setup

We evaluate on RoboMME (Dai et al., 2026), a benchmark of 16 long-horizon manipulation tasks that each place a different demand on memory. We group them by their six functional characteristics: Motion-Centric (tracking a trajectory), Time-Sensitive (acting at the correct moment), Short- and Long-Horizon Video (grounding objects from a short or an extended video), Dynamic Scene-Change (adapting to a scene that changes during execution), and Event-Salient (progress marked by discrete events). All models are trained on the RoboMME demonstrations of all 16 tasks in a multi-task setting: the SigLIP encoder is frozen, and the Gemma VLM, the action expert, and the parameters a memory adds (for D&R, the selector) are fine-tuned with the flow-matching action loss. The policy predicts a chunk of 20 actions and executes the first 16. Every model is trained for 80k steps with a batch size of 64 (TTT uses 16); implementation details and results by the benchmark's four suites are in Appendix B.

Baselines. To answer what information the memory should carry, we compare against the three memory families: perceptual memory, TokenDrop and FrameSamp (Dai et al., 2026); latent memory, HAMLET (Koo et al., 2026) and RB-VLA (Bagaria et al., 2026); and parametric memory, TTT (Sun et al., 2025). All memory methods build on the $\pi_{0.5}$ backbone and are modulated into the action expert in the same way (Figure 3(b)). Our code, including the $\pi_{0.5}$ re-implementations of HAMLET and RB-VLA and all training configurations, will be released upon acceptance.

Evaluation protocol. All memory methods are given the same budget of 64 tokens and differ only in what fills it. Each model is evaluated on all 16 tasks with 50 episodes per task (800 in total), on environment seeds disjoint from training and a maximum horizon of 1,300 steps. Results are averaged over the last three checkpoints and three random seeds (nine runs in total).

6.2 Experimental results

Main results. Overall, D&R reaches the highest average success rate over the four suites (38.6%, against 32.2% for HAMLET, 27.9% for FrameSamp, and 17.9% for the memoryless $\pi_{0.5}$; Table 7). The memory visualisations show that D&R retrieves the key fact: in Figure 5, for example, it keeps the grasp pose from the MoveCube video at the start of the episode, more than 300 steps earlier. Latent memory collapses on Imitation (19.5% for HAMLET and 10.2% for RB-VLA, which scores

Table 1: Per-task success rate (%) on RoboMME (Dai et al., 2026), tasks grouped by functional characteristic (the same results by suite are in Table 7). All memories are built on $\pi_{0.5}$ with a budget of 64 tokens and differ in what they store: raw patch tokens kept by frame sampling (FrameSamp) or token dropping (TokenDrop); learned moment tokens (HAMLET) or a recursive belief (RB-VLA); or fast weights (TTT). Each number is the mean over the last three checkpoints and three seeds; standard deviations are in Table 7. Avg is the mean over the tasks of a characteristic; **bold** is the highest success rate in each column.

		Motion-Centric					Time-Sensitive	Short-Horizon Video				Long-Horizon Video			Dynamic Scene-Change				Event-Salient			
Type	Method	Pattern Lock	Route Stick	Swing Xtimes	Insert Peg	Avg	Stop Cube	Video Umsk	Video Repick	Video UmskS	Avg	Video PlcBtn	Video PlcOrd	Avg	Button Umsk	Button UmskS	Pick HighL	Avg	Pick Xtimes	Bin Fill	Move Cube	Avg
No memory	$\pi_{0.5}$	2.89	4.67	35.56	1.56	11.17	6.67	20.44	0.44	18.67	13.18	31.11	25.78	28.45	22.22	6.67	11.33	13.41	42.89	30.00	26.00	32.96
Perceptual	FrameSamp	15.56	17.78	88.45	2.22	31.00	27.34	30.00	14.22	20.89	21.70	26.89	25.56	26.23	6.67	5.11	18.67	10.15	60.44	34.44	52.66	49.18
	TokenDrop	9.56	8.89	36.67	2.44	14.39	5.78	28.67	1.56	16.67	15.63	26.67	19.56	23.12	16.67	6.44	14.00	12.37	39.78	27.78	24.44	30.67
Latent	HAMLET	7.33	9.56	81.11	1.11	24.78	**39.33**	27.78	20.67	25.11	24.52	27.78	21.78	24.78	26.67	**21.78**	26.00	24.82	**84.67**	34.89	**60.00**	**59.85**
	RB-VLA	0.67	0.20	86.00	**3.33**	22.55	10.67	30.00	26.00	23.33	26.44	24.67	24.00	24.34	**28.00**	20.00	**26.67**	**24.89**	79.33	**36.67**	36.67	50.89
Parametric	TTT	4.67	7.11	41.56	0.89	13.56	2.89	28.67	4.44	19.78	17.63	20.44	17.11	18.77	15.11	3.33	12.44	10.29	42.67	28.22	24.67	31.85
Ours	D&R	**32.00**	**59.33**	**95.33**	**3.33**	**47.50**	31.33	**32.67**	**26.67**	**27.33**	**28.89**	**34.00**	**29.33**	**31.67**	26.00	21.33	24.00	23.78	82.67	35.33	56.67	58.22

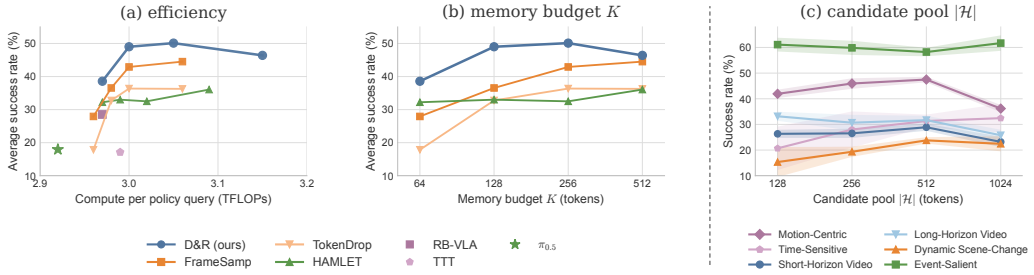


Figure 6: (a) Average success rate against the TFLOPs of one policy forward pass; each line sweeps the memory budget K from 64 to 512 (Table 8). (b) Average success rates against K. (c) D&R per task characteristic (Dai et al., 2026) as the candidate pool $|\mathcal{H}|$ grows from 128 to 1024 tokens at $K = 64$; mean $\pm$ std over runs (Table 5).

0.0% on RouteStick): a compressed state was unable to adequately preserve the required spatial detail. More results are in Appendix B.3.

To understand where each memory helps, we report the results by the functional characteristics of the tasks (Table 1) rather than by the four suites, which name cognitive memory types. We categorise the historical key fact into frame-like, one past frame that the current observation points to (Motion-Centric, Short- and Long-Horizon Video), and state-like, the running state of a process that unfolds during the episode, such as elapsed time, a changing scene, or a count of events (Time-Sensitive, Dynamic Scene-Change, Event-Salient); the two kinds separate the memory methods clearly. The key observations are: (i) D&R gains the most on *frame-like facts*: 47.5% on Motion-Centric against 31.0% for FrameSamp and 24.8% for HAMLET, and on Long-Horizon Video it is the only memory above the memoryless policy (31.7% against 28.5%). (ii) Latent memory is competitive on *state-like facts*: HAMLET leads Time-Sensitive (39.3% against 31.3%) and ties D&R on Dynamic Scene-Change and Event-Salient (24.8 and 59.9 against 23.8 and 58.2), where RB-VLA also ties on the former (24.9). A latent memory updated at every step tracks a running state by construction, whereas D&R has to reconstruct it from snapshots. For example, in BinFill, it learns to keep one snapshot of each cube dropped into the bin (Figure 9).

On InsertPeg, every method stays at or below 3.3%, against 1.6% for the memoryless $\pi_{0.5}$. The rollout in Figure 22 reaches the correct end of the peg and the correct side of the box, so the failure lies in the precision of the insertion rather than in memory.

Efficiency against performance. Memory buys sufficiency with computation. Varying the budget from 64 to 512 tokens and measuring cost as the TFLOPs of one forward pass at the average episode length (Figure 6(a)), D&R reaches the highest success at essentially the same cost per query. RB-VLA, by contrast, updates its belief recurrently at every step, so training cannot be parallelised over time and takes about 10 days for 80k steps, whereas D&R takes about 20 hours (Appendix B).

Real-world robot manipulation. We test whether memory keeps a real robot reliable over several long-run tasks, with a person intervening mid-task and under perception noise (Dey et al., 2026). On four tasks, PutBottles, TrackCube, RepickCube and DrawPattern, D&R succeeds in 35 of 40 trials, against 22 for FrameSamp and 3 for $\pi_{0.5}$ (Table 2), with the largest margin on PutBottles, the longest task at up to 4 minutes. FrameSamp fails when its memory holds the wrong frame, and errors accumulate over time, with most failures on the second cube of RepickCube and the third move of DrawPattern. Without memory, $\pi_{0.5}$ falls back on a fixed behaviour, starting eight of its nine failed DrawPattern trials with the same leftward move whatever the video showed. Details are in Appendix B.4; videos of all 40 D&R trials are in the supplementary material.

Table 2: Real-world results.

Method	Put Bottles	Track Cube	Repick Cube	Draw Pattern	Total
$\pi_{0.5}$	0/10	2/10	0/10	1/10	3/40
FrameSamp	4/10	8/10	6/10	4/10	22/40
D&R	**9/10**	**9/10**	**9/10**	**8/10**	**35/40**

6.3 Ablation

Candidate pool size. D&R does not need to read the full history: the selector works on a candidate pool of $|\mathcal{H}|$ tokens, $|\mathcal{H}|/16$ frames sampled uniformly from the history with 16 pooled patch tokens each, which keeps its cost independent of the growing history length. We sweep $|\mathcal{H}|$ from 128 to 1024 at the same budget $K = 64$ (Tables 4 and 5). The key observations are: state-like facts grow with the pool (20.7% to 32.4% on Time-Sensitive), since covering more of the time axis helps track the process, whereas frame-like facts need specific tokens retrieved at full fidelity, and a larger pool hides them among more candidates, so the selector is harder to learn; Motion-Centric peaks at 512 (47.5%) and drops at 1024 (36.2%).

Which sufficient statistic to learn. To ablate how the training objective (the sufficient statistic) shapes what the memory builds up, we add D&R+pred, which trains D&R with RB-VLA's world-model loss on top of the action loss (Appendix B.2); D&R, D&R+pred, and RB-VLA then span policy sufficiency, sufficiency plus prediction, and a fully predictive belief. At a pool of $|\mathcal{H}| = 256$ (Table 6), the prediction loss brings no gain: asking the memory to predict the future spends capacity that the action does not buy back (Section 4).

Historical token types. Adding the instruction's text tokens to the candidate pool (+text in Table 6) costs 6.6 points overall and 14.5 on imitation. Our main experiments therefore use patch tokens only. Action tokens are excluded from the candidates because of causal confusion: the action-conditioned policy might learn to "cheat" by simply outputting the previous action, which is often highly (yet spuriously) correlated with the next action.

Effect of the memory budget. Figure 6(b) plots the average success rate as the memory budget grows from 64 to 512 tokens. D&R leads at every budget, but its curve flattens at 256 tokens (50.1%) and dips at 512 (46.4%), which we attribute to convergence: a larger budget needs more training steps, and at the 80k steps reported here the 512-token model is still improving.

7 Conclusion, Limitations, and Future Work

In this paper, we proposed Divide-and-Remember (D&R), a recursive memory method that composes a single lightweight selector over the history to maximise the conditional mutual information $I(a_t; m_t \mid o_t)$ between the action and the memory in the pretrained token space. Experimental results showed that D&R achieved state-of-the-art success rates across different memory demands, and that the gain transferred to a real robot.

On state-like facts D&R only ties latent memory, and on StopCube and ButtonUnmask it falls behind, since D&R stores and uses history tokens directly, whereas a latent memory can reformulate or aggregate knowledge in a more compact way. A potential next step is to train the memory to re-value the selected tokens so as to store information more compactly, which could extend its capacity.

To keep the comparison fair, every memory method was trained in the same multi-task setting without any task-specific information. As a result, the memory modules may converge to acquiring the knowledge shared across tasks, thereby leading to redundancy. There is still design space for a memory method that can enhance the parameter efficiency and also retain specialisation by focusing on distinctive aspects.

AI USE STATEMENT

In this work, we used generative AI tools for editing and checking the manuscript, for literature search, and for refining code. We have not used generative AI tools for research ideation, for generating the method implementation, or for producing experimental data or results; the remaining categories are not applicable to this work. We have reviewed all AI-assisted work, every derivation was checked by the authors, and all text was read and edited by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Rishabh Agarwal, Marlos C. Machado, Pablo Samuel Castro, and Marc G Bellemare. Contrastive Behavioral Similarity Embeddings for Generalization in Reinforcement Learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=qda7-sVg84.

Stefano V. Albrecht and Subramanian Ramamoorthy. Exploiting causality for selective belief filtering in dynamic bayesian networks. *J. Artif. Int. Res.*, 55(1):1135–1178, January 2016. ISSN 1076-9757.

Vaidehi Bagaria, Bijo Sebastian, and Nirav Kumar Patel. Recursive Belief Vision Language Action Models. *arXiv preprint arXiv:2602.20659*, 2026.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 27–30 Sep 2025.

Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic Markov Decision Processes. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20)*, 2020.

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. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=8m30ogkPk2.

Sharmita Dey, Robert Riener, Strahinja Dosen, and Stefano V. Albrecht. From autonomy to alliance: Robotic foundation models must learn with us, not just for us. *Science Robotics*, 11(113): eaea1822, 2026. doi: 10.1126/scirobotics.aea1822. URL https://www.science.org/doi/abs/10.1126/scirobotics.aea1822.

Mhairi Dunion, Trevor McInroe, Kevin Sebastian Luck, Josiah P. Hanna, and Stefano V Albrecht. Conditional Mutual Information for Disentangled Representations in Reinforcement Learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=EmYWJsyad4.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite Markov decision processes. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, pp. 162–169, Arlington, Virginia, USA, 2004. AUAI Press. ISBN 0974903906.

Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence*, 147(1):163–223, 2003. ISSN 0004-3702. Planning with Uncertainty and Incomplete Information.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, 1998. ISSN 0004-3702.

Myungkyu Koo, Daewon Choi, Taeyoung Kim, Kyungmin Lee, Changyeon Kim, Younggyo Seo, and Jinwoo Shin. HAMLET: Switch Your Vision-Language-Action Model into a History-Aware Policy. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=KcJ9U0x6kO.

Wouter Kool, Herke van Hoof, and Max Welling. Stochastic Beams and Where To Find Them: The Gumbel-Top-k Trick for Sampling Sequences Without Replacement. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pp. 3499–3508, 2019.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

Hao Li, Shuai Yang, Yilun Chen, Xinyi Chen, Xiaoda Yang, Yang Tian, Hanqing Wang, Tai Wang, Dahua Lin, Feng Zhao, and Jiangmiao Pang. Towards efficient and robust manipulation via multi-frame vision-language-action modeling. In *Proceedings of the Fortieth AAAI Conference on Artificial Intelligence and Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence and Sixteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'26/IAAI'26/EAAI'26. AAAI Press, 2026. ISBN 978-1-57735-906-7.

Tianwei Ni, Michel Ma, Benjamin Eysenbach, and Pierre-Luc Bacon. When Do Transformers Shine in RL? Decoupling Memory from Credit Assignment. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

David L. Poole and Alan K. Mackworth. *Artificial Intelligence: Foundations of Computational Agents*. Cambridge University Press, 3 edition, 2023.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. DynamicViT: Efficient Vision Transformers with Dynamic Token Sparsification. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. MemoryVLA: Perceptual-Cognitive Memory in Vision-Language-Action Models for Robotic Manipulation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=54U3XHf7qq.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (Learn at Test Time): RNNs with Expressive Hidden States. In *Forty-second International Conference on Machine Learning*, 2025.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

Ganlin Yang, Zhangzheng Tu, Yuqiang Yang, Sitong Mao, Junyi Dong, Tianxing Chen, Jiaqi Peng, Jing Xiong, Jiafei Cao, Jifeng Dai, et al. EventVLA: Event-Driven Visual Evidence Memory for Long-Horizon Vision-Language-Action Policies. *arXiv preprint arXiv:2606.20092*, 2026.

Xuehui Yu, Mhairi Dunion, Xin Li, and Stefano V Albrecht. Skill-aware Mutual Information Optimisation for Zero-shot Generalisation in Reinforcement Learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=GtbwJ6mruI.

Yihan Zeng, Minghao Ye, Yiyuan Chen, Yide Shentu, Philipp Wu, Zike Yan, and Zhongyu Li. KEMO: Event-Driven Keyframe Memory for Long-Horizon Robot Manipulation with VLA Policies. *arXiv preprint arXiv:2606.23589*, 2026.

Yifeng Zhu, Abhishek Joshi, Peter Stone, and Yuke Zhu. VIOLA: Object-Centric Imitation Learning for Vision-Based Robot Manipulation. In *CoRL*, pp. 1199–1210, 2022. URL https://proceedings.mlr.press/v205/zhu23a.html.

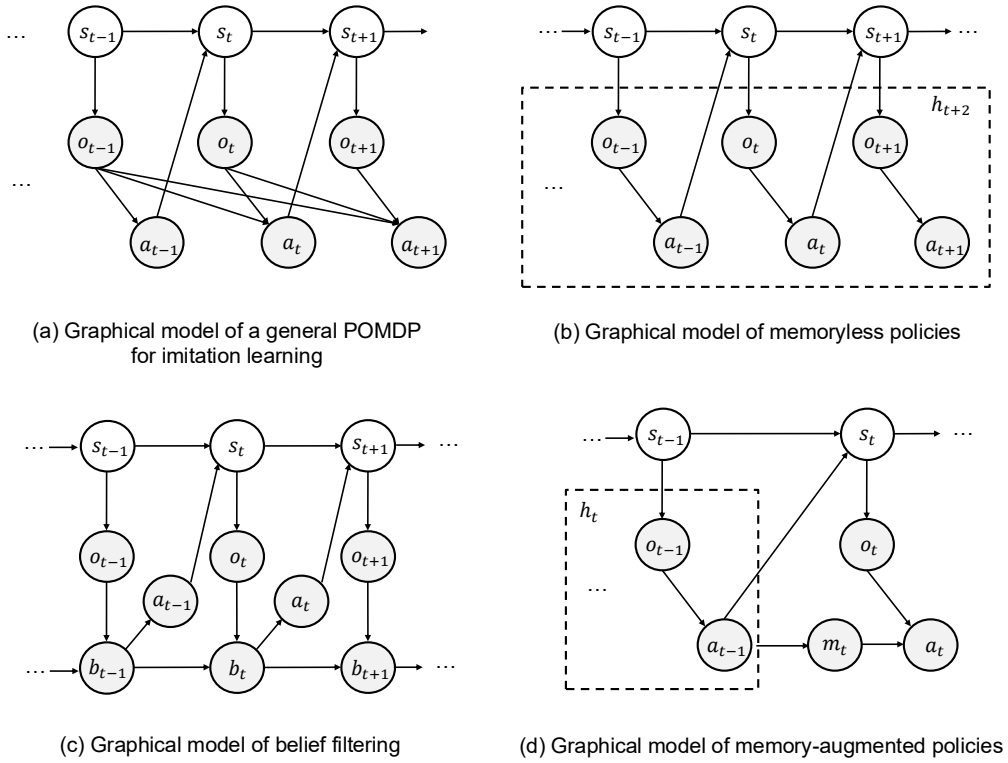


Figure 7: **Graphical models of imitation learning in a POMDP**. (a) A general policy for the POMDP inputs the current observation and the full history, $a_t \sim \pi(\cdot \mid o_t, h_t)$, but the history grows without bound. (b) A memoryless policy $a_t \sim \pi(\cdot \mid o_t)$ inputs only the current observation, which causes the memory problem in Figure 1(a). (c) A belief policy $a_t \sim \pi(\cdot \mid b_t)$ inputs the belief $b_t(s) = P(s_t = s \mid h_t, o_t)$, the posterior over the current state and a sufficient statistic of the history. (d) A memory-augmented policy $a_t \sim \pi(\cdot \mid o_t, m_t)$ inputs a memory $m_t = \mathcal{M}(h_t)$ and the current observation. Here m_t need only be sufficient to reproduce the expert's action (Proposition 1), and the optimal memory carries the most action-relevant information $I(a_t;\, m_t \mid o_t)$. Shaded circles are observed in the demonstrations, white circles are hidden.

A From first principles of POMDP theory

Existing memory methods either select part of the history by an inductive bias (frame novelty, a hand-designed rule, or a verifier), or learn the memory through an auxiliary loss, e.g., predicting the next state. Both report gains, which leaves open how to choose between them, so we return to the first principles of partially observable Markov decision processes (POMDPs), in which the agent receives only partial or noisy observations of the state before it acts (Figure 7(a)).

With $h_t = \{(o_k, a_k)\}_{k<t}$ the observations and actions up to time step t, one solution is to condition the policy on the whole history, $\pi(\cdot \mid o_t, h_t)$, by solving the associated dynamic decision network (Poole & Mackworth, 2023). However, the number of t-step histories, and hence the size of such a policy, grows exponentially in t, so this works only for short histories. A common alternative is a compact statistic of the history. The belief state, the distribution over states given the history (Figure 7(c)), is a sufficient statistic and turns the POMDP into an MDP over beliefs (Kaelbling et al., 1998) (Section A.1), but its dimension grows with the number of states. We therefore seek a representation of the history that scales to the high-dimensional observations and large models of VLAs (Section A.2).

A.1 BELIEF

Belief is a sufficient statistic. Recall that $h_t = \{(o_k, a_k)\}_{k<t}$ contains the observation–action pairs before time t, while o_t is the current observation. The belief state b_t (Kaelbling et al., 1998) specifies the probability of the agent being in each state given all information available before choosing a_t, $b_t(s) = P(s_t = s \mid h_t, o_t)$.

Definition 2 (sufficient history representation)**.** A statistic $x_t = \phi(h_t, o_t)$ of the available history and current observation is *sufficient* for the state if

$$P\big(s_t = s \mid h_t, o_t\big) = P\big(s_t = s \mid x_t\big) \qquad \forall s \in \mathcal{S},\ \forall (h_t, o_t), \tag{13}$$

i.e., x_t determines the posterior over s_t (Kaelbling et al., 1998).

The belief state satisfies Equation 13 by construction, so it is a sufficient statistic of the available history and current observation (Albrecht & Ramamoorthy, 2016). Unfortunately, the dimension of a belief state grows linearly with the number of states, so for a large (e.g. continuous) state space we want to learn a more compact representation.

Kaelbling et al. (1998) show that a belief state can be computed recursively,

$$\begin{aligned} b_{t+1}(s_{t+1}) &= P\big(s_{t+1} \mid h_{t+1}, o_{t+1}\big) \\ &= P\big(s_{t+1} \mid h_t, o_t, a_t, o_{t+1}\big) \\ &\propto P\big(o_{t+1} \mid s_{t+1}\big)\, P\big(s_{t+1} \mid h_t, o_t, a_t\big) \\ &= P\big(o_{t+1} \mid s_{t+1}\big) \sum_{s_t} P\big(s_{t+1} \mid s_t, a_t\big)\, P\big(s_t \mid h_t, o_t\big), \end{aligned} \tag{14}$$

where the last line uses the Markov property of the dynamics and the definition of b_t. The recursion shows that b_{t+1} depends on the past only through b_t, a_t and o_{t+1}. More generally, the state posterior represented by any sufficient x_t can be updated from (x_t, a_t, o_{t+1}), as in a recurrent neural network.

Normalising Equation 14 gives the **belief filtering** update (Albrecht & Ramamoorthy, 2016): for discrete hidden states,

$$b_{t+1}(s_{t+1}) = \frac{p(o_{t+1} \mid s_{t+1}) \sum_{s_t \in \mathcal{S}} p(s_{t+1} \mid s_t, a_t)\, b_t(s_t)}{\eta}, \tag{15}$$

and for continuous hidden states,

$$b_{t+1}(s_{t+1}) = \frac{p(o_{t+1} \mid s_{t+1}) \int p(s_{t+1} \mid s_t, a_t)\, b_t(s_t)\, \mathrm{d}s_t}{\eta}, \tag{16}$$

where η normalises the numerator over $s_{t+1} \in \mathcal{S}$. The two factors play different roles: the transition $p(s_{t+1} \mid s_t, a_t)$ propagates the belief through the dynamics alone and carries no new evidence, while the observation function $p(o_{t+1} \mid s_{t+1})$ reweights that prediction by the incoming observation.

Bisimulation for measuring the sufficiency of a belief state. Bisimulation metrics measure state similarity in reinforcement learning by *behavioural equivalence*, looking at transitions and rewards rather than raw state features. The concept originated in model checking and MDP minimisation (Givan et al., 2003): two states are aggregated if they yield the same immediate reward and the same distribution over next equivalent states. Approximate bisimulation metrics (Ferns et al., 2004; Castro, 2020) relax the exact equivalence into a distance, extending the idea to high-dimensional environments.

Definition 3 (π-bisimulation metric (Castro, 2020))**.** Given two states $s_i, s_j \in \mathcal{S}$, the π-bisimulation metric d_π measures their behavioural similarity as the fixed point of

$$d_\pi(s_i, s_j) = \underbrace{\big|R^\pi_{s_i} - R^\pi_{s_j}\big|}_{(A)\ \text{reward sufficiency}} + \underbrace{\gamma\, W_1(d_\pi)\big(P^\pi_{s_i}, P^\pi_{s_j}\big)}_{(B)\ \text{transition sufficiency}}, \tag{17}$$

where R^π is the reward under policy π, P^π the transition distribution, γ the discount factor, and W_1 the Wasserstein distance.

A useful tool for a policy that generalises is knowing which states lead to similar behaviour and which do not; ideally the similarity looks beyond the immediately chosen action to long-term behaviour, which the π-bisimulation metric does through the full sequence of future rewards. Grounding the metric in rewards, however, can be too restrictive (the policies agree but the obtained rewards do not) or too permissive (the policies differ but the rewards do not) (Agarwal et al., 2021). In imitation learning, moreover, no reward is available at all; we instead assume behaviour policies $\pi^\star$ are optimal.

The policy similarity metric (PSM) (Agarwal et al., 2021) therefore replaces the reward difference in Equation 17 with a probability pseudometric DIST between the policies themselves, keeping the same recursive second term: states are close when the policies at these states, and at their future states, are similar (Agarwal et al. (2021) ground the metric in the optimal policy). For a given DIST, the PSM d^* satisfies the recursive equation

$$d^*(s_i, s_j) = \underbrace{\mathrm{DIST}\big(\pi(s_i), \pi(s_j)\big)}_{(A)\text{ policy sufficiency}} + \underbrace{\gamma\, W_1(d^*)\big(P^\pi_{s_i}, P^\pi_{s_j}\big)}_{(B)\text{ transition sufficiency}}, \tag{18}$$

where the discount factor sets the weight between the two terms, and d^* is finite whenever DIST is bounded. The policy-sufficiency term measures the instantaneous behaviour; the transition-sufficiency term measures the long-term effect carried through the transition dynamics.

A.2 Memory

Instead of the belief state, we consider another representation of the history for visuomotor policies.

Memory is a policy-sufficient statistic. We incorporate memory $m_t = \mathcal{M}(h_t)$ into the graphical model of the POMDP for memory-augmented policies (Figure 7(d)), and seek $\mathcal{M}$ such that (o_t, m_t) d-separates the expert action a_t from the remaining history h_t, equivalently $I(a_t; h_t \mid o_t, m_t) = 0$. The memory is written deterministically from the history, $p(m_t \mid h_t) = \delta_{\mathcal{M}(h_t)}(m_t)$; unlike a belief, it is not required to predict the next observation, $p(o_{t+1} \mid m_t, a_t)$, only to render a_t independent of h_t given (o_t, m_t).

We measure what information the memory should carry:

Proposition (Proposition 1, restated)**.** The information that the history h_t carries for control beyond the current observation is the conditional mutual information $I\big(a_t; h_t \mid o_t\big) = \mathbb{E}_{p(h_t, o_t)}[D_{\mathrm{KL}}\big(\pi^\star(\cdot \mid o_t, h_t) \,\big\|\, p(\cdot \mid o_t)\big)]$, the expected divergence between the expert policy and the action distribution obtained by marginalising it over the history. Let the memory be any function of the full history, $m_t = \mathcal{M}(h_t)$, inserted into the graphical model of Figure 7(d). By the chain rule, the memory splits this information into the part it keeps and the part it discards,

$$I\big(a_t; h_t \mid o_t\big) \;=\; I\big(a_t; m_t \mid o_t\big) \;+\; I\big(a_t; h_t \mid o_t, m_t\big).$$

The left-hand side of Equation 3 is fixed by the task, so the **optimal memory for visuomotor policies** needs to keep the most action-relevant information and discard the least,

$$\arg\max_{\mathcal{M}} \; I\big(a_t; m_t \mid o_t\big).$$

A memory that maximises Equation 4 is a policy-sufficient statistic (Figure 8), i.e., $\pi^\star(a_t \mid o_t, h_t) = \pi^\star(a_t \mid o_t, m_t)$ for all (o_t, h_t).

The belief is a sufficient statistic of the history: every reward function and every future observation can be predicted from it. A memory need only be sufficient for the policy.

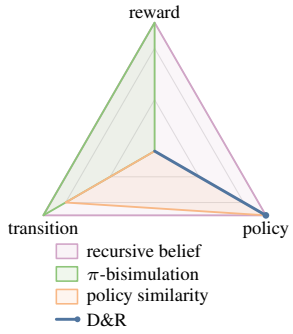


Figure 8: Different methods assume different sufficiency metrics on the history. Bisimulation measures reward and transition; policy similarity measures policy and transition; the belief takes all three. D&R keeps policy sufficiency alone, under a limited budget.

To see what can actually be optimised, we unfold the memory term of Proposition 1:

$$\begin{aligned} I\big(a_t; m_t \mid o_t\big) &= H(a_t \mid o_t) \;-\; H\big(a_t \mid o_t, m_t\big) && \text{(definition)} \\ &= \mathbb{E}_{\mathcal{D}}\big[\log p(a_t \mid o_t, m_t)\big] + H(a_t \mid o_t) && \text{(expand the residual)} \\ &\geq\; \mathbb{E}_{\mathcal{D}}\big[\log \pi_\theta(a_t \mid o_t, m_t)\big] + H(a_t \mid o_t) && \text{(variational } \pi_\theta \text{ for } p) \end{aligned}$$

The first two lines are written in terms of the true $p(a_t \mid o_t, m_t)$ of the demonstration distribution and cannot be evaluated directly; replacing it by the policy π_θ gives a variational lower bound, and $H(a_t \mid o_t)$ is a constant of the task. Maximising the memory term is therefore lower-bounded by minimising the action loss,

$$\max_{\mathcal{M}} \; I\big(a_t;\, m_t \mid o_t\big) \;\; \geq \;\; \max_{\mathcal{M},\,\theta} \; \mathbb{E}_{\mathcal{D}}\big[\,\log \pi_\theta(a_t \mid o_t, m_t)\big] + H(a_t \mid o_t),$$

Policy and memory can thus be trained together end-to-end against a single action loss, with no auxiliary objective and no supervision on what to remember.

Memory under the policy similarity metric. We revisit the policy similarity metric of Equation 18, taking DIST to be the total variation distance. A memory is a compressed history, so we measure it against the history it compresses:

$$d^*(m_t, h_t) = \underbrace{\mathrm{TV}\big(\pi^\star(\cdot \mid o_t, m_t),\; \pi^\star(\cdot \mid o_t, h_t)\big)}_{(A)\text{ policy sufficiency}} \;+\; \underbrace{W_1(d^*)\big(P^{\pi^\star}_{o_t,m_t},\; P^{\pi^\star}_{h_t}\big)}_{(B)\text{ transition sufficiency}}, \tag{19}$$

where we drop γ: in RL it is given by the discount factor, and LfD has none. Term (A) is related to the quantity Proposition 1 optimises, and term (B) is left out.

B EXPERIMENTAL DETAILS

B.1 EXPERIMENTAL SETUP

We follow a unified training recipe across all methods to ensure a fair comparison. Specifically, we adopt the released $\pi_{0.5}$ configuration from the RoboMME benchmark and exclude proprioceptive states from the inputs, as they bring no benefit. To accelerate training, we freeze the SigLIP vision backbone and precompute and cache all visual tokens, fine-tuning only the VLM expert, the action expert, and any parameters introduced by the memory (for D&R, the selector). All models are trained for 80K steps; the perceptual and latent memory variants use a batch size of 64, while the parametric (TTT) variants use 16. All models are trained on $4\times$A800 GPUs with cached data. Training takes about 20 hours for the perceptual and latent memory variants, 45 hours for TTT, and 10 days for RB-VLA. Detailed hyperparameters are provided in Table 3.

Table 3: Training hyperparameters.

Hyperparameter	**Value**
Base model	$\pi_{0.5}$ base
Batch size	64 global
Action horizon	20
Action space	Joint-space
Proprioception	False
Total steps	80,000
Warmup steps	10,000
LR	5×10^{-5}
LR schedule	Constant
Optimiser	AdamW (β_1=0.9, β_2=0.95, weight decay 0)
Gradient clip norm	1.0
EMA decay	0.999
Frozen part	SigLIP ViT
Hardware	$4\times$ NVIDIA A800, 80 GB
Parallelism	FSDP across the 4 GPUs

B.2 VARIANTS OF D&R

D&R+pred. This variant adds RB-VLA's world-model objective (Bagaria et al., 2026) to D&R, leaving the selector, the reduction tree and the policy unchanged. For every frame f of the candidate pool we form a frame latent $z_f \in \mathbb{R}^{512}$: the mean of the frame's patch tokens after the memory encoder, RMS-normalised and linearly projected. Two predictors map the two-frame context $[z_{f-1}, z_f]$ and a recency feature of frame f to the next latent and to the latent h pool frames ahead,

$$\mathcal{L}_{\text{wm}} = \mathbb{E}_f\left[\|\hat{z}_{f+1} - \text{sg}(z_{f+1})\|^2\right] + \tfrac{1}{2}\,\mathbb{E}_f\left[\|\hat{z}_{f+h} - \text{sg}(z_{f+h})\|^2\right], \tag{20}$$

where sg is the stop-gradient and the expectation runs over valid (non-padded) frame pairs; we use $h = 4$ on the 16-frame pool. The two predictors are two-layer MLPs with SiLU activations, and the pool budget, the selector, and the policy are identical to D&R. The total loss is $\mathcal{L}_{\text{action}} + \mathcal{L}_{\text{wm}}$, and the gradient of $\mathcal{L}_{\text{wm}}$ reaches the memory encoder through the predictors' inputs, so the memory features are asked to be predictive of the future as well as sufficient for the action. The stop-gradient targets and the $1{:}\frac{1}{2}$ weighting of the one-step and h-step terms follow RB-VLA; we omit its inverse-dynamics and KL terms, which require per-step actions and a stochastic belief that a static candidate pool does not have.

Table 4 sweeps the size of the candidate pool (Table 5 regroups the same runs by task characteristic), and Figure 6(c) plots the same runs per task characteristic; Table 6 ablates the training signal and the candidate set at a fixed pool of $|\mathcal{H}| = 256$, against D&R at the same pool size.

Table 4: **Candidate pool.** Success rate (%) of D&R averaged over the four tasks of each suite, as mean ± standard deviation over evaluation runs (checkpoints × seeds $0/7/42$), as the size $|\mathcal{H}|$ of the candidate pool the selector reduces grows from 8 to 64 frames. Every row uses a memory budget of $K = 64$ tokens. **bold** is the highest success rate in each column.

Variant	Counting	Permanence	Reference	Imitation	AVG
$\lvert\mathcal{H}\rvert = 128$ (8 frames)	58.25 ± 2.04	20.83 ± 4.79	27.00 ± 1.38	34.67 ± 1.70	35.19 ± 1.98
$\lvert\mathcal{H}\rvert = 256$ (16 frames)	59.17 ± 2.84	25.17 ± 2.47	24.56 ± 2.13	$\mathbf{38.67 \pm 1.99}$	36.89 ± 0.81
$\lvert\mathcal{H}\rvert = 512$ (32 frames)	61.17 ± 0.62	$\mathbf{26.83 \pm 1.03}$	$\mathbf{28.50 \pm 1.41}$	37.83 ± 2.78	$\mathbf{38.58 \pm 0.50}$
$\lvert\mathcal{H}\rvert = 1024$ (64 frames)	$\mathbf{61.89 \pm 2.37}$	26.17 ± 2.32	20.94 ± 1.57	28.67 ± 2.58	34.42 ± 1.03

Table 5: **Candidate pool by task characteristic.** Success rate (%) of D&R averaged over the tasks of each characteristic of Dai et al. (2026) (the grouping of Table 1), as mean ± standard deviation over evaluation runs (checkpoints × seeds $0/7/42$), as the size $|\mathcal{H}|$ of the candidate pool changes. Every row uses a memory budget of $K = 64$ tokens; the AVG column is that of Table 4. **bold** is the highest success rate in each column.

Variant	Motion-Centric	Time-Sensitive	Short-Horizon Video	Long-Horizon Video	Dynamic Scene-Change	Event-Salient	AVG
$\lvert\mathcal{H}\rvert = 128$ (8 frames)	41.92 ± 1.62	20.67 ± 8.46	26.33 ± 1.58	$\mathbf{33.17 \pm 1.77}$	15.33 ± 5.84	61.11 ± 2.69	35.19 ± 1.98
$\lvert\mathcal{H}\rvert = 256$ (16 frames)	45.94 ± 2.05	28.00 ± 7.06	26.52 ± 1.80	30.67 ± 3.06	19.33 ± 1.99	59.85 ± 2.77	36.89 ± 0.81
$\lvert\mathcal{H}\rvert = 512$ (32 frames)	$\mathbf{47.50 \pm 1.08}$	31.33 ± 2.49	$\mathbf{28.89 \pm 1.37}$	31.67 ± 1.70	$\mathbf{23.78 \pm 1.26}$	58.22 ± 1.66	$\mathbf{38.58 \pm 0.50}$
$\lvert\mathcal{H}\rvert = 1024$ (64 frames)	36.17 ± 1.41	$\mathbf{32.44 \pm 5.56}$	23.19 ± 2.06	25.78 ± 2.15	22.44 ± 2.95	$\mathbf{61.70 \pm 3.02}$	34.42 ± 1.03

B.3 Additional simulation results

B.3.1 Full results on all tasks

We report results by suite: counting, permanence, reference and imitation, which ask the policy to accumulate a count over repeated events, to track an object after it is hidden, to refer back to an earlier moment of the episode, and to reproduce a demonstrated sequence. Table 7 reports every method by suite and by task, with standard deviations over seeds; the main text groups the same numbers by task characteristic (Table 1).

Table 6: **Ablations.** Success rate (%) of D&R and its variants at a candidate pool of $|\mathcal{H}| = 256$, as mean $\pm$ standard deviation over evaluation runs (checkpoints $\times$ seeds $0/7/42$), averaged over the four tasks of each suite (top) and per task (bottom). +pred adds RB-VLA's world-model loss to the action loss; +text adds the language instruction to the candidate pool the selector scores. Every row uses a memory budget of $K = 64$ tokens. **bold** is the highest success rate in each column.

Variant	Counting	Permanence	Reference	Imitation	AVG
D&R (reference)	**59.17 ± 2.84**	**25.17 ± 2.47**	**24.56 ± 2.13**	**38.67 ± 1.99**	**36.89 ± 0.81**
+pred	58.62 ± 1.85	22.69 ± 1.52	22.12 ± 1.43	30.75 ± 2.96	33.55 ± 0.70
+text	52.00 ± 2.31	21.94 ± 3.51	23.11 ± 2.02	24.22 ± 3.15	30.32 ± 1.54

	Counting				Permanence			
Variant	BinFill	PickXtimes	SwingXtimes	StopCube	VideoUnmask	ButtonUnmask	VideoUnmaskSwap	ButtonUnmaskSwap
D&R (reference)	38.44 ± 4.79	78.67 ± 4.00	91.56 ± 3.74	**28.00 ± 7.06**	**35.11 ± 6.74**	24.22 ± 4.57	**24.89 ± 3.41**	16.44 ± 6.24
+pred	**41.25 ± 4.47**	**80.00 ± 5.00**	**94.75 ± 2.44**	18.50 ± 5.81	31.00 ± 3.74	**25.75 ± 3.23**	15.75 ± 4.41	**18.25 ± 2.73**
+text	38.44 ± 2.95	68.89 ± 8.60	83.78 ± 6.76	16.89 ± 4.01	34.00 ± 3.27	19.78 ± 5.77	22.67 ± 3.13	11.33 ± 6.32

	Reference				Imitation			
Variant	PickHighlight	VideoRepick	VideoPlaceButton	VideoPlaceOrder	MoveCube	InsertPeg	PatternLock	RouteStick
D&R (reference)	**17.33 ± 4.32**	19.56 ± 3.24	**39.78 ± 5.12**	21.56 ± 4.30	**62.44 ± 6.02**	**4.67 ± 1.33**	**32.00 ± 7.72**	**55.56 ± 3.74**
+pred	15.50 ± 4.21	19.25 ± 3.73	27.00 ± 3.16	**26.75 ± 2.82**	50.25 ± 2.73	1.75 ± 1.85	27.50 ± 7.33	43.50 ± 3.97
+text	16.22 ± 4.57	**21.11 ± 4.01**	32.22 ± 3.94	22.89 ± 5.74	50.89 ± 6.33	1.11 ± 0.99	20.22 ± 4.66	24.67 ± 4.62

Table 7: **Full results on RoboMME.** Success rate (%) of every method under the 64-token budget, grouped by what the memory stores. Top: mean over the four tasks of each suite and over all 16 tasks (AVG). Bottom: every task, as mean $\pm$ standard deviation over seeds. The $\pi_{0.5}$ row is the number reported by Dai et al. (2026). **bold** is the highest success rate in each column.

Type	Method	**Counting**	**Permanence**	**Reference**	**Imitation**	**AVG**
No memory	$\pi_{0.5}$	28.78	17.00	17.17	8.78	17.93
Perceptual	FrameSamp	52.67	15.67	21.34	22.06	27.93
	TokenDrop	27.50	17.11	15.45	11.33	17.85
Latent	HAMLET	60.00	25.34	24.06	19.50	32.22
	RB-VLA	53.17	25.33	25.34	10.22	28.52
Parametric	TTT	28.84	16.72	13.61	9.34	17.13
Ours	D&R	**61.17**	**26.83**	**28.50**	**37.83**	**38.58**

		Counting				Permanence			
Type	Method	BinFill	PickXtimes	SwingXtimes	StopCube	VideoUnmask	ButtonUnmask	VideoUnmaskSwap	ButtonUnmaskSwap
No memory	$\pi_{0.5}$	30.00 ± 6.48	42.89 ± 6.41	35.56 ± 5.27	6.67 ± 5.20	20.44 ± 2.96	22.22 ± 10.22	18.67 ± 4.24	6.67 ± 4.12
Perceptual	FrameSamp	34.44 ± 2.57	60.44 ± 3.10	88.45 ± 3.00	27.34 ± 7.36	30.00 ± 3.31	6.67 ± 2.18	20.89 ± 1.57	5.11 ± 2.20
	TokenDrop	27.78 ± 4.29	39.78 ± 6.82	36.67 ± 5.57	5.78 ± 5.33	28.67 ± 2.83	16.67 ± 6.48	16.67 ± 3.61	6.44 ± 4.67
Latent	HAMLET	34.89 ± 3.48	**84.67 ± 6.86**	81.11 ± 6.86	**39.33 ± 6.78**	27.78 ± 2.73	26.67 ± 3.32	25.11 ± 1.76	**21.78 ± 6.28**
	RB-VLA	**36.67 ± 5.03**	79.33 ± 2.31	86.00 ± 5.29	10.67 ± 4.62	30.00 ± 3.46	**28.00 ± 3.46**	23.33 ± 1.15	20.00 ± 4.00
Parametric	TTT	28.22 ± 6.82	42.67 ± 4.80	41.56 ± 5.46	2.89 ± 2.67	28.67 ± 4.69	15.11 ± 8.31	19.78 ± 2.33	3.33 ± 4.36
Ours	D&R	35.33 ± 0.94	82.67 ± 1.89	**95.33 ± 3.77**	31.33 ± 2.49	**32.67 ± 1.89**	26.00 ± 0.30	**27.33 ± 3.40**	21.33 ± 4.11

		Reference				Imitation			
Type	Method	PickHighlight	VideoRepick	VideoPlaceButton	VideoPlaceOrder	MoveCube	InsertPeg	PatternLock	RouteStick
No memory	$\pi_{0.5}$	11.33 ± 4.90	0.44 ± 0.88	31.11 ± 4.01	25.78 ± 1.86	26.00 ± 3.61	1.56 ± 1.94	2.89 ± 2.26	4.67 ± 2.45
Perceptual	FrameSamp	18.67 ± 2.37	14.22 ± 1.13	26.89 ± 1.66	25.56 ± 3.86	52.66 ± 0.94	2.22 ± 1.37	15.56 ± 0.83	17.78 ± 1.13
	TokenDrop	14.00 ± 3.74	1.56 ± 1.33	26.67 ± 3.00	19.56 ± 4.33	24.44 ± 4.56	2.44 ± 2.19	9.56 ± 2.96	8.89 ± 3.02
Latent	HAMLET	26.00 ± 4.24	20.67 ± 2.65	27.78 ± 5.33	21.78 ± 2.73	**60.00 ± 4.80**	1.11 ± 1.45	7.33 ± 2.24	9.56 ± 2.96
	RB-VLA	**26.67 ± 9.45**	26.00 ± 2.00	24.67 ± 4.16	24.00 ± 2.00	36.67 ± 1.15	**3.33 ± 1.15**	0.67 ± 1.15	0.20 ± 0.01
Parametric	TTT	12.44 ± 3.13	4.44 ± 2.79	20.44 ± 4.45	17.11 ± 9.49	24.67 ± 4.58	0.89 ± 1.45	4.67 ± 2.24	7.11 ± 2.26
Ours	D&R	24.00 ± 1.63	**26.67 ± 0.94**	**34.00 ± 0.40**	**29.33 ± 3.40**	56.67 ± 4.71	**3.33 ± 0.94**	**32.00 ± 5.89**	**59.33 ± 3.40**

B.3.2 Ablation: memory budget

Table 8 reports the average success rate as the memory budget grows from 64 to 512 tokens, overall and per suite.

Table 8: **Effect of the memory budget.** Average success rate (%) on RoboMME as the memory budget grows from 64 to 512 tokens, overall and per suite. The FrameSamp entries from 128 to 512 tokens and the TokenDrop entries at 128 and 256 tokens are the numbers reported by Dai et al. (2026); the remaining entries are our runs.

	Overall			
Budget	FrameSamp	TokenDrop	HAMLET	D&R
64	27.93	17.85	32.22	38.58
128	36.54	32.76	33.00	49.01
256	42.90	36.36	32.50	50.11
512	44.51	36.27	36.06	46.40

Table 9 repeats the per-task comparison of Table 7 with a memory budget of 512 tokens.

Table 9: **Results with a memory budget of** 512 **tokens.** Per-task success rate (%) on RoboMME (Dai et al., 2026) as mean ± standard deviation over seeds, with $\pi_{0.5}$ as the backbone and methods grouped by what the memory stores. The $\pi_{0.5}$, FrameSamp, and TTT rows are the numbers reported by Dai et al. (2026). **bold** is the highest success rate in each column.

		Counting				Permanence			
Type	Method	BinFill	PickXtimes	SwingXtimes	StopCube	VideoUnmask	ButtonUnmask	VideoUnmaskSwap	ButtonUnmaskSwap
No memory	$\pi_{0.5}$	30.00 ± 6.48	42.89 ± 6.41	35.56 ± 5.27	6.67 ± 5.20	20.44 ± 2.96	22.22 ± 10.22	18.67 ± 4.24	6.67 ± 4.12
Perceptual	FrameSamp	39.56 ± 5.27	87.33 ± 2.45	**92.00 ± 2.24**	42.00 ± 11.36	**32.67 ± 3.16**	25.11 ± 3.18	24.44 ± 6.06	18.22 ± 3.80
	TokenDrop	31.33 ± 6.02	84.67 ± 4.84	86.00 ± 4.56	11.00 ± 4.69	30.67 ± 2.07	23.67 ± 4.46	22.67 ± 5.01	21.00 ± 6.03
Latent	HAMLET	31.78 ± 2.91	**88.89 ± 3.48**	89.33 ± 3.74	**52.67 ± 7.42**	29.78 ± 3.23	25.11 ± 4.26	23.33 ± 3.32	22.67 ± 3.61
	RB-VLA	**39.67 ± 3.13**	78.31 ± 5.01	81.00 ± 8.50	12.02 ± 1.15	30.01 ± 3.21	24.20 ± 2.66	23.65 ± 0.13	19.10 ± 3.00
Parametric	TTT	34.89 ± 6.57	63.78 ± 6.12	41.33 ± 9.95	4.00 ± 3.46	31.78 ± 6.12	22.44 ± 3.13	19.56 ± 6.98	18.00 ± 6.86
Ours	**D&R**	38.00 ± 1.63	68.67 ± 0.94	**92.00 ± 4.32**	48.67 ± 0.94	26.67 ± 2.49	**26.00 ± 1.63**	**30.67 ± 1.89**	**30.00 ± 5.89**

		Reference				Imitation			
Type	Method	PickHighlight	VideoRepick	VideoPlaceButton	VideoPlaceOrder	MoveCube	InsertPeg	PatternLock	RouteStick
No memory	$\pi_{0.5}$	11.33 ± 4.90	0.44 ± 0.88	31.11 ± 4.01	25.78 ± 1.86	26.00 ± 3.61	1.56 ± 1.94	2.89 ± 2.26	4.67 ± 2.45
Perceptual	FrameSamp	22.89 ± 3.89	30.44 ± 5.81	**60.00 ± 4.00**	32.00 ± 3.87	77.78 ± 3.80	7.56 ± 3.57	**53.56 ± 4.56**	**66.67 ± 4.12**
	TokenDrop	21.00 ± 5.02	25.67 ± 4.63	45.67 ± 4.27	30.33 ± 4.63	70.33 ± 5.85	3.67 ± 2.94	21.67 ± 3.44	51.00 ± 6.29
Latent	HAMLET	24.67 ± 6.00	25.11 ± 2.85	34.00 ± 2.45	21.56 ± 2.96	75.33 ± 7.00	6.22 ± 3.07	8.00 ± 3.46	18.22 ± 2.33
	RB-VLA	24.37 ± 9.35	23.01 ± 4.01	23.77 ± 5.01	23.01 ± 2.10	38.61 ± 1.10	2.31 ± 1.05	0.61 ± 1.30	0.01 ± 0.30
Parametric	TTT	12.22 ± 3.80	9.56 ± 3.43	34.00 ± 6.00	22.89 ± 4.59	33.56 ± 6.06	0.89 ± 1.05	3.11 ± 2.85	5.56 ± 3.28
Ours	**D&R**	**33.33 ± 1.89**	**52.00 ± 5.89**	54.00 ± 1.63	**34.67 ± 2.49**	**86.00 ± 2.83**	**7.67 ± 0.47**	53.33 ± 2.49	60.67 ± 3.40

B.3.3 Memory visualisation on all tasks

Figure 5 in the main text visualises the memory of one MoveCube episode. Figures 9–24 repeat the visualisation for every RoboMME task.

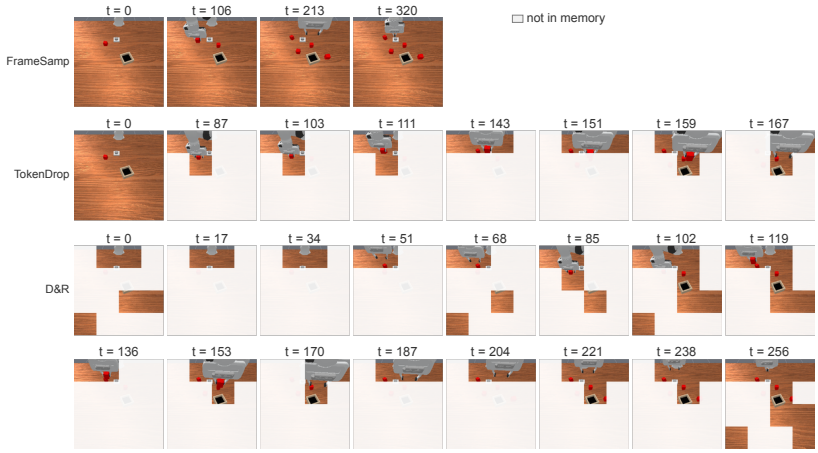


Figure 9: **BinFill** (Counting): "put one red cube into the bin, then press the button to stop".

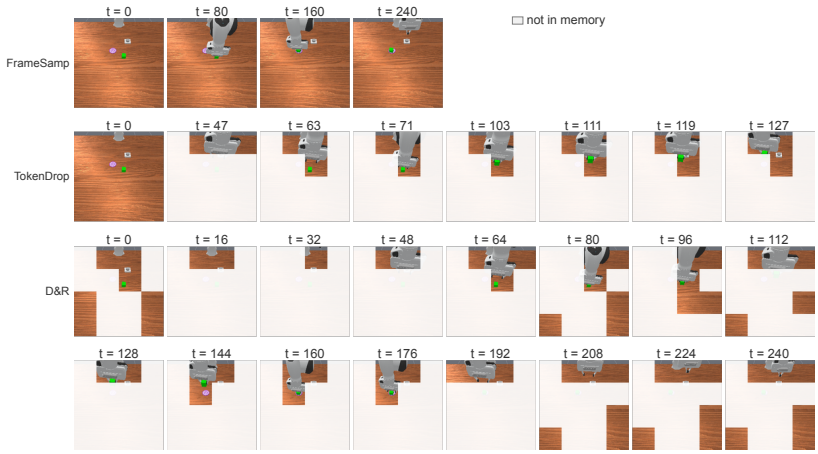


Figure 10: **PickXtimes** (Counting): "pick up the green cube and place it on the target, then press the button to stop".

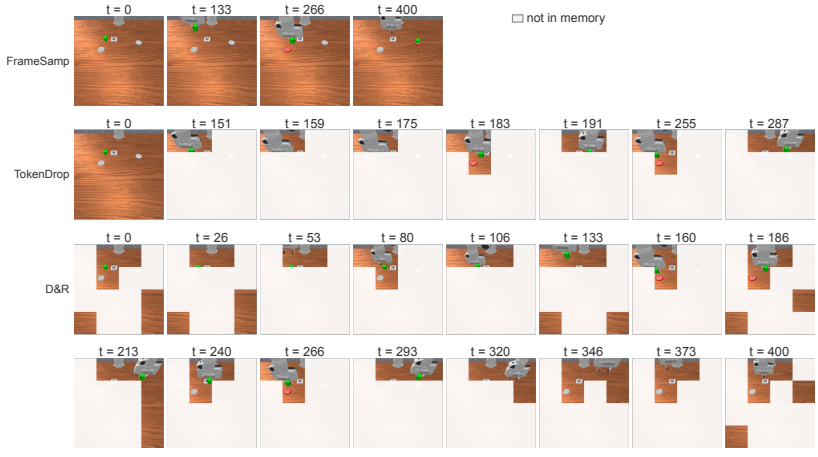


Figure 11: **SwingXtimes** (Counting): "pick up the green cube, move it to the top of the right-side target, then move it to the top of the left-side target, repeating this back-and-forth motion two times, finally press the button to stop".

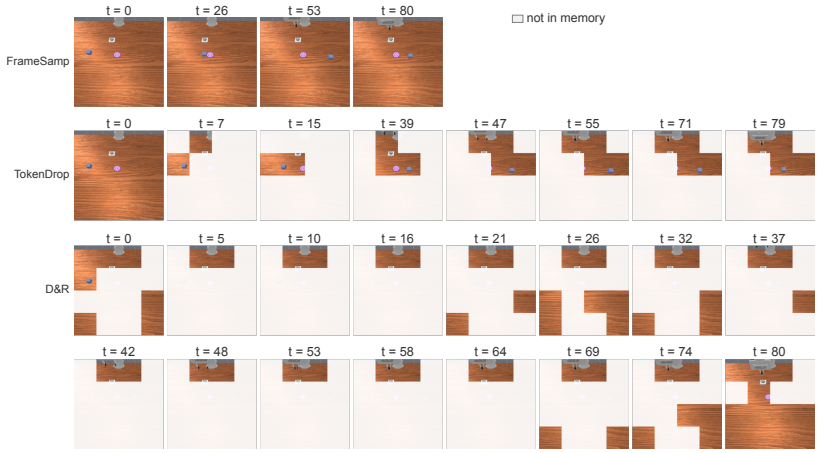


Figure 12: **StopCube** (Counting): "press the button to stop the cube just as it reaches the target for the fourth time".

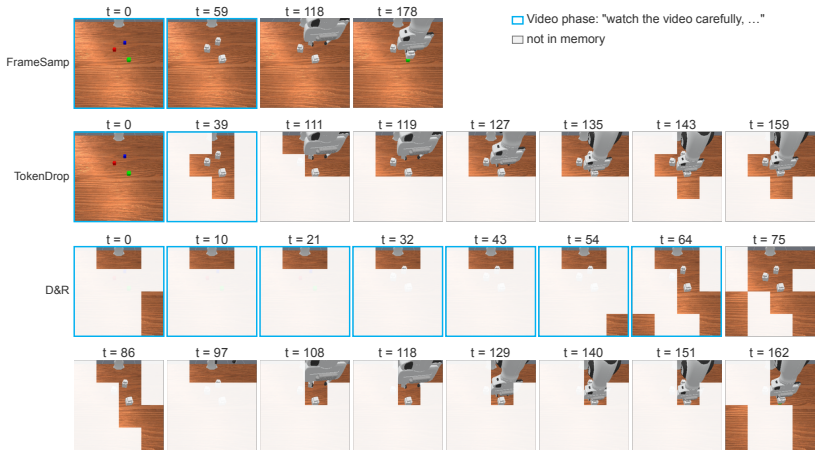


Figure 13: **VideoUnmask** (Permanence): "watch the video carefully, then pick up the container hiding the green cube".

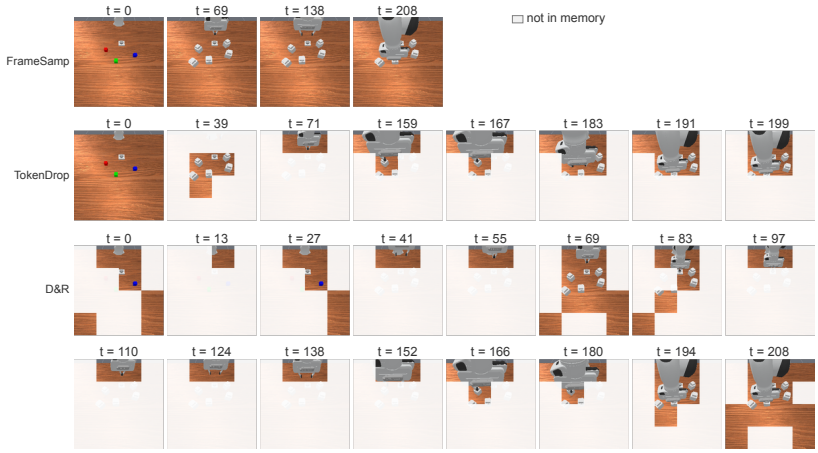


Figure 14: **ButtonUnmask** (Permanence): "first press the button, then pick up the container hiding the green cube".

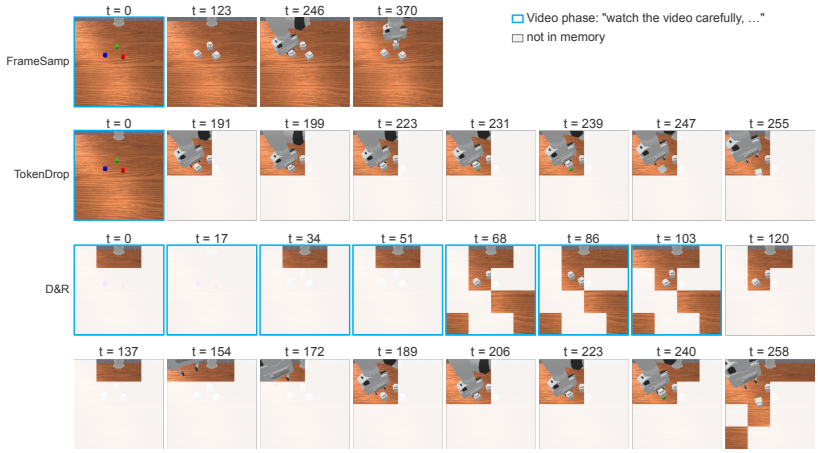


Figure 15: **VideoUnmaskSwap** (Permanence): "watch the video carefully, then pick up the container hiding the green cube".

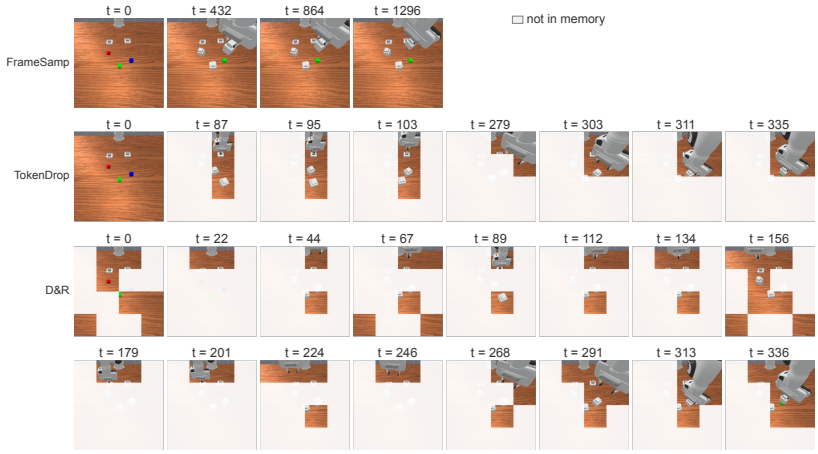


Figure 16: **ButtonUnmaskSwap** (Permanence): "first press both buttons on the table, then pick up the container hiding the green cube".

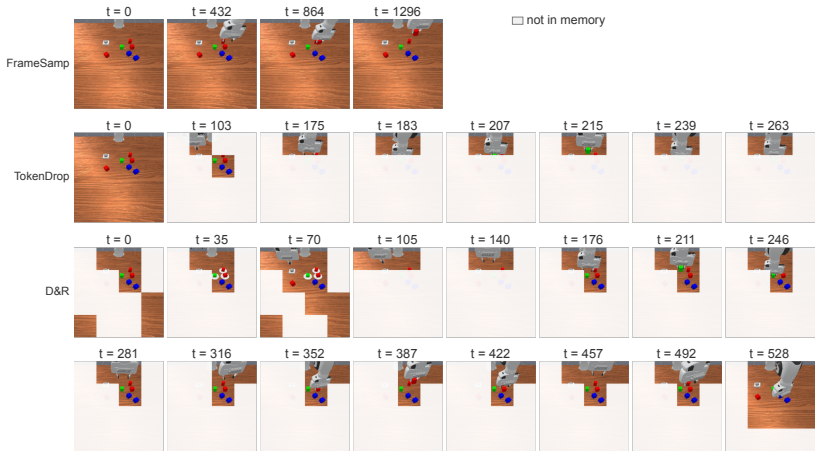


Figure 17: **PickHighlight** (Reference): "first press the button, then pick up all cubes that have been highlighted with white areas on the table".

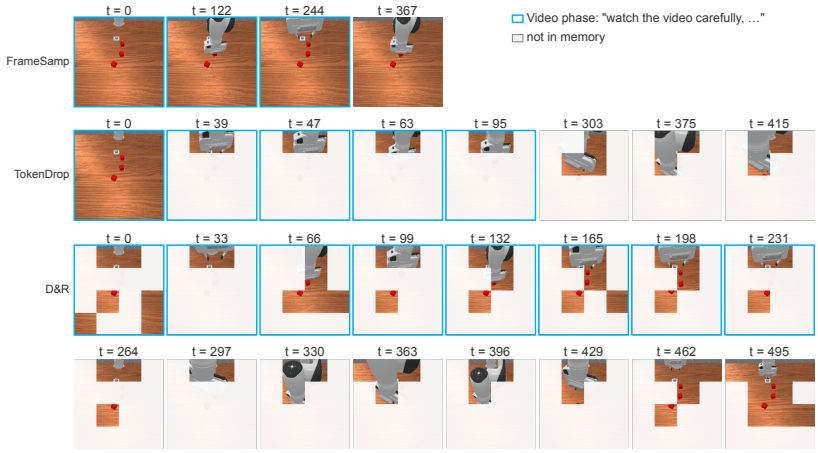


Figure 18: **VideoRepick** (Reference): "watch the video carefully, then pick up the same block that was previously picked up again, finally put it down and press the button to stop".

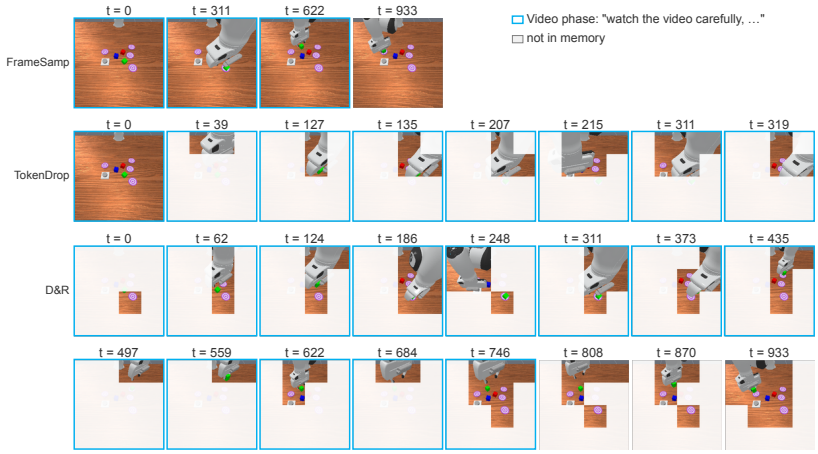


Figure 19: **VideoPlaceButton** (Reference): "watch the video carefully, then place the green cube on the target right before the button was pressed".



Figure 20: **VideoPlaceOrder** (Reference): "watch the video carefully, then place the green cube on the second target it was previously placed on".

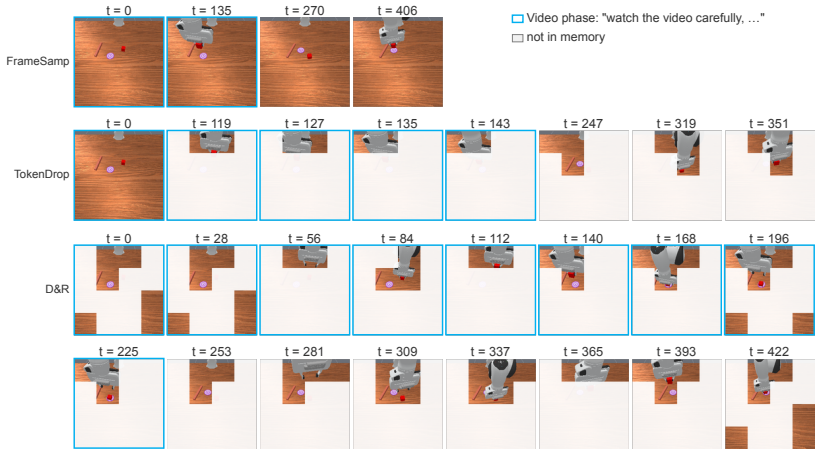


Figure 21: **MoveCube** (Imitation): "watch the video carefully, then move the cube to the target in the same manner as before".

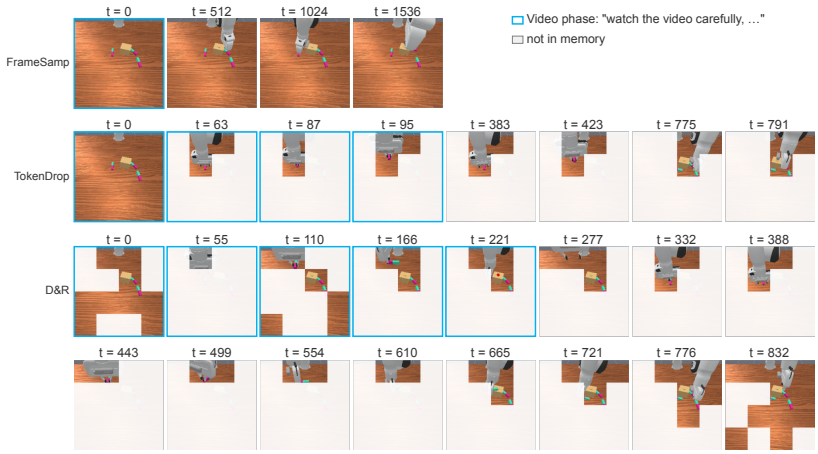


Figure 22: **InsertPeg** (Imitation): "watch the video carefully, then grasp the same end of the same peg you've picked before and insert it into the same side of the box".

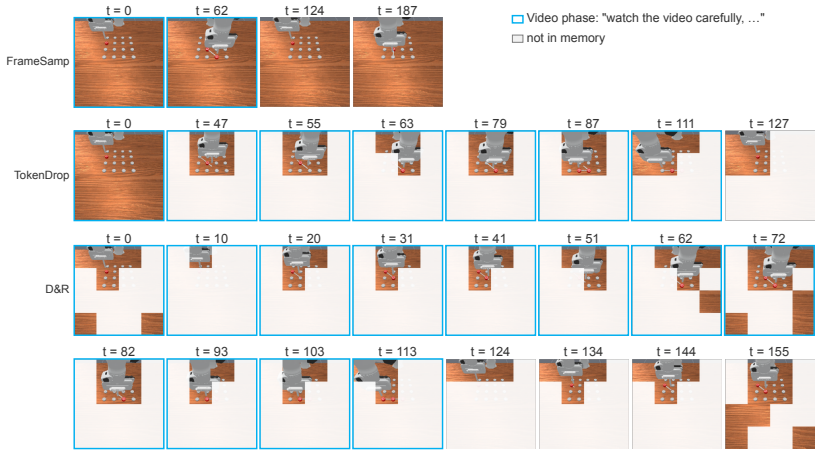


Figure 23: **PatternLock** (Imitation): "watch the video carefully, then use the stick attached to the robot to retrace the same pattern".

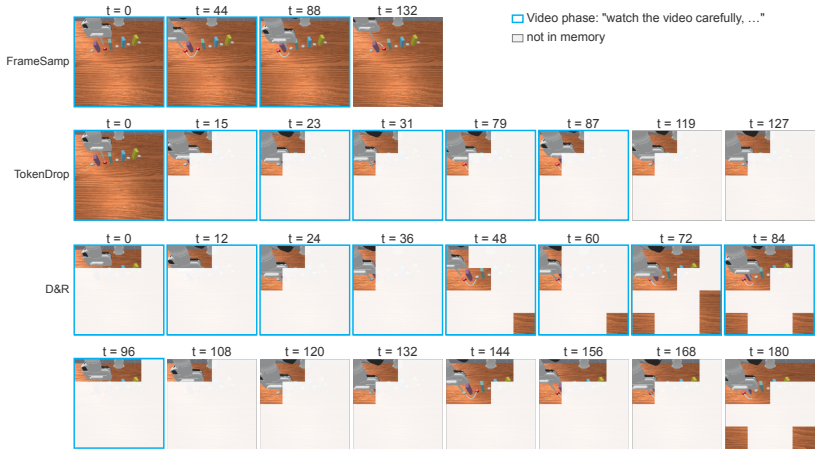


Figure 24: **RouteStick** (Imitation): "watch the video carefully, then use the stick attached to the robot to navigate around the sticks on the table, following the same path".

B.4 REAL ROBOT EXPERIMENT

Following Dai et al. (2026), we evaluate D&R on a real robot with four tasks, PutBottles, TrackCube, RepickCube, and DrawPattern, which mirror the simulation tasks BinFill, VideoUnmask, VideoRepick, and PatternLock, respectively (Figure 25).

- PutBottles (Counting). The robot transfers $N \in \{1, 2\}$ bottles from a basket to a bin. During execution, a human may intervene by removing placed bottles or adding new ones, so the robot cannot rely on the current observation alone and must track its progress over time.
- TrackCube (Permanence). Given a pre-recorded video in which a human hides cubes under identical cups, the robot must lift the cup that hides the cube of the named colour; $N \in \{3, 4\}$ is the number of cups (one cube under one of three cups, or three cubes under three of four cups).
- RepickCube (Reference). Given a pre-recorded video in which a human picks up two of the $N \in \{3, 4\}$ cubes on the table, the robot must pick up the same two cubes.
- DrawPattern (Imitation). Given a pre-recorded video in which a human traces a path of $N \in \{2, 3, 4\}$ moves on a 3×3 grid, the robot must reproduce the same path.

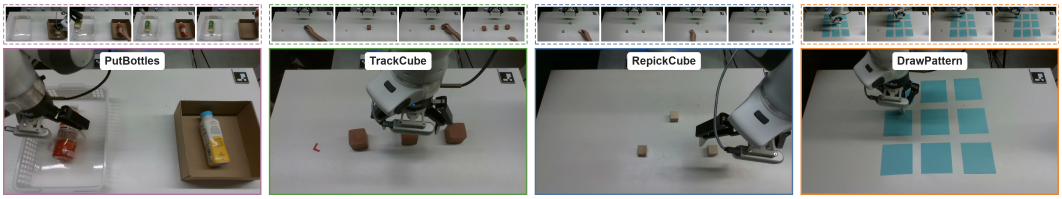


Figure 25: The four real-world tasks, recorded on our Franka Panda: PutBottles (Counting), TrackCube (Permanence), RepickCube (Reference) and DrawPattern (Imitation). For each task, the upper dashed box shows frames from the earlier history (the video the policy watched; for PutBottles, earlier frames of the episode, including a human intervention), and the lower box shows the front-camera observation during execution.

Protocol. We use a 7-DoF Franka Emika Panda arm with a Robotiq 2F-140 gripper, controlled through deoxys (Zhu et al., 2022) in joint-position mode at 1000 Hz, and two Intel RealSense cameras: a D435i mounted on the wrist and a D435 on a tripod in front of the workspace (Figure 26). The memory is built from the front view only; the wrist view is fed to the policy as a current observation. Demonstrations are collected by teleoperation at 15 Hz, 100 per task, and, as in simulation, all methods are trained in a multi-task learning setting on the four tasks. Each task is evaluated over 10 rollouts per method; we compare $\pi_{0.5}$, FrameSamp with a budget of $K = 2048$ tokens, and D&R with $K = 1024$ tokens drawn from a pool of $|\mathcal{H}| = 2048$. The budgets are larger than in simulation because the real-robot episodes are longer. All other hyperparameters are unchanged, except that training runs for 50k steps. Table 2 reports the results.

Results. Table 2 reports the successes over the 10 trials per task. We inspected every failed trial using the rollout videos and the robot logs (gripper width, grasp poses and stop reasons).

(1) PutBottles. Without memory, $\pi_{0.5}$ cannot count. It overshoots in four trials (a further grasp after the last bottle), stops one bottle short in two, and never transfers a bottle in three. FrameSamp remembers the wrong count in three trials, stops short in one and never transfers a bottle in one. In one trial each, $\pi_{0.5}$ and FrameSamp place both bottles for N=2 but the trial is still judged a failure. D&R fails once, grasping a third bottle after placing both for N=2.

(2) TrackCube. All eight failures of $\pi_{0.5}$ uncover a wrong cup. Both FrameSamp failures do the same, because the memory holds the wrong frame. D&R lifts the correct cup in all ten trials; one is logged as a failure because the run was stopped before the automatic put-back.

(3) RepickCube. $\pi_{0.5}$ never picks the right cubes, and two trials end in a safety stop. Three of FrameSamp's four failures pick the first cube correctly and then fail from the wrong memory, heading for a wrong cube or stopping; the fourth grasps no cube. The only D&R failure picks the first cube and then a wrong second one.

(4) DrawPattern. $\pi_{0.5}$ makes the same leftward move in eight of its nine failed trials, whatever the demonstration, and four of them end with a safety stop. FrameSamp reproduces the first two



Figure 26: Real-robot platform. A 7-DoF Franka Emika Panda with a Robotiq 2F-140 gripper and two RealSense cameras: a wrist camera mounted on the gripper and a front camera on a tripod facing the workspace. The objects of the four tasks (baskets, cubes, cups, and bottles) are laid out on the table.

moves, but as the error in its memory accumulates it goes wrong at the third in four trials; in the other two it adds a move at the end or ends imprecisely. The two D&R failures come from a wrong movement direction, at the third move in one trial and at the first in the other.