ELMUR: External Layer Memory with Update/Rewrite for Long-Horizon RL

Egor Cherepanov **Alexey K. Kovalev** **Aleksandr I. Panov**
AIRI. Moscow, Russia MIPT, Dolgoprudny, Russia
{cherepanov,kovalev,panov}@airi.net

Abstract: Real-world robotic agents must act under partial observability and long horizons, where key cues may appear long before they affect decision making. However, most modern approaches rely solely on instantaneous information, without incorporating insights from the past. Standard recurrent or transformer models struggle with retaining and leveraging long-term dependencies: context windows truncate history, while naive memory extensions fail under scale and sparsity. We propose **ELMUR** (**E**xternal **L**ayer **M**emory with **U**pdate/**R**ewrite), a transformer architecture with structured external memory. Each layer maintains memory embeddings, interacts with them via bidirectional cross-attention, and updates them through an **L**east **R**ecently **U**sed **(LRU)** memory module using replacement or convex blending. ELMUR extends effective horizons up to 100,000 times beyond the attention window and achieves a 100% success rate on a synthetic T-Maze task with corridors up to one million steps. In POPGym, it outperforms baselines on more than half of the tasks. On MIKASA-Robo sparse-reward manipulation tasks with visual observations, it nearly doubles the performance of strong baselines. These results demonstrate that structured, layer-local external memory offers a simple and scalable approach to decision making under partial observability.

Keywords: Memory, Robotics, POMDP

1 Introduction

Imagine a robot cooking pasta: it stirs once, adds salt, and later adds salt again, repeating until the dish is inedible. The issue is simple: the robot cannot remember if salt was already added, since it dissolves invisibly, nor how much is still in the container. This is a case of partial observability — the world rarely reveals all necessary information. Humans recall past actions effortlessly, but robots lack this ability. Though effective in controlled settings [1, 2], robots often fail under partial observability [3, 4]. Standard recurrent [5] and transformer [6] models rely heavily on short observation windows, making them brittle under long-horizon dependencies and sparse signals. This motivates hybrid memory-augmented transformers that explicitly store and retrieve past information [3, 7].

Within the Reinforcement Learning (RL) paradigm [8], long-horizon challenges are compounded by sample inefficiency and sparse rewards: real-world exploration is costly and unsafe, while simulation suffers from a sim-to-real gap [9]. Offline RL mitigates this with pre-collected datasets [10], but usually assumes dense feedback; reshaping sparse rewards demands domain knowledge and risks bias [11, 12, 13]. In robotics, delayed feedback makes long-term memory indispensable. A complementary paradigm is Imitation Learning (IL) [14], whose simplest form, Behavior Cloning (BC), reduces control to supervised learning on demonstration pairs. Building on this idea, recent Vision-Language-Action (VLA) models [15, 16, 1] scale with large datasets, yet their fixed transformer windows [3, 7] leave three challenges: (i) extending context without quadratic cost, (ii) mitigating truncation-induced forgetting, and (iii) retaining task-relevant information across long horizons. This motivates our central question: **how can we equip IL policies with efficient long-term memory to solve long-horizon, partially observable tasks?**

To address these challenges, we introduce **ELMUR** (**E**xternal **L**ayer **M**emory with **U**pdate/**R**ewrite), a transformer architecture in which every layer is augmented with a structured external layer memory (Figure 1). ELMUR combines three ingredients: (i) layer-local memory embeddings that

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

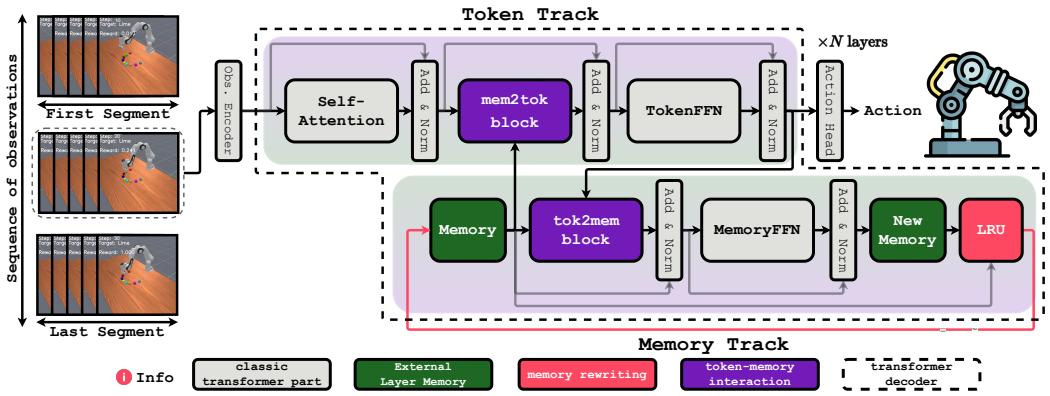


Figure 1: **ELMUR overview.** Each transformer layer is augmented with an external memory track that runs in parallel with the token track. Tokens attend to memory through a `mem2tok` block, while memory embeddings are updated from tokens through a `tok2mem` block. LRU block selectively rewrites memory via replacement or convex blending, ensuring bounded yet persistent storage. This design enables token-memory interaction and long-horizon recall beyond the attention window.

persist across segments, (ii) bidirectional token–memory read/write interaction via cross-attention (`mem2tok`, `tok2mem`), and (iii) a Least Recently Used (LRU) update block that refreshes memory through replacement or convex blending, balancing stability and adaptability. This design enables efficient segment-level recurrence and extends retention of task-relevant information up to $100{,}000\times$ beyond the native attention window, making long-horizon decision making feasible in robotics.

We evaluate ELMUR on the synthetic T-Maze [17], the robotic MIKASA-Robo [4] suite of sparse-reward manipulation tasks with visual observations, and the diverse POPGym benchmark [18], all designed to test memory under partial observability. ELMUR achieves a 100% success rate on T-Maze corridors up to one million steps, nearly doubles baseline performance on MIKASA-Robo, and obtains the top score on 24 of 48 POPGym tasks. These results demonstrate that ELMUR enables stable retention of task-relevant information, efficient long-term storage, and robust generalization under partial observability.

Our contributions are twofold:

- **We propose ELMUR**, a transformer with layer-local external memory, bidirectional token-memory cross-attention, and an LRU-based update rule rewriting memory via replacement or convex blending (Section 3). This design extends memory horizons far beyond the attention window.
- **We empirically demonstrate** that ELMUR achieves robust generalization under partial observability across synthetic, robotic, and puzzle/control tasks (Section 5).
- **We provide a theoretical analysis** of LRU-based memory dynamics, establishing formal bounds on forgetting, retention horizons, and stability of memory embeddings (Section 4).

2 Background

Many real-world robotic and control tasks involve partial observability, where the agent cannot directly access the true system state [19]. This setting is modeled as a partially observable Markov decision process (POMDP), defined as the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, Z, R, \rho_0, \gamma)$, with latent state space $\mathcal{S}$, action space $\mathcal{A}$, and observation space $\mathcal{O}$. The transition dynamics are $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, where $T(s' \mid s, a)$ is the probability of reaching s' after taking a in s. The observation function $Z : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{O})$ specifies $Z(o \mid s', a)$, the probability of observing o after reaching s' under action a. The reward function is $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, the initial state distribution is $\rho_0 \in \Delta(\mathcal{S})$, and $\gamma \in (0, 1)$ is the discount factor.

In the special case of full observability, the observation equals the state ($o_t \equiv s_t$), reducing the POMDP to a Markov decision process (MDP). The optimal policy then depends only on the current state, $\pi^*(a_t \mid s_t)$. In the general POMDP case, however, the agent cannot access s_t directly and must rely on the full history $h_t = (o_0, a_0, o_1, a_1, \ldots, o_t)$, yielding $\pi^*(a_t \mid h_t)$. A practical alternative is to approximate history with a learned memory state $m_t = f_\phi(m_{t-1}, o_t, a_{t-1}), \quad \pi_\theta : \mathcal{M} \to \Delta(\mathcal{A}), \quad \pi_\theta(a_t \mid m_t)$, where f_ϕ is a, for instance, recurrent [20] or memory-augmented [21] update rule.

Algorithm 1 ELMUR layer update for segment i at layer ℓ. Inputs are token hidden states $h \in \mathbb{R}^{B\times L\times d}$, memory (m, p) with $m \in \mathbb{R}^{B\times M\times d}$, anchors $p \in \mathbb{Z}^{B\times M}$, and absolute times t. Outputs are updated hidden states h' and memory (m', p').

// Input embedding (before first layer) – to encode observation
$h \leftarrow \mathrm{ObsEncoder}(o)$
// Token track – sequence processing and enrichment with information from memory
$h \leftarrow \mathrm{AddNorm}\big(h + \mathrm{SelfAttention}(h;\ \text{causal mask})\big)$
$B_{\text{rel}} \leftarrow \mathrm{RelativeBias}(t, p)$ *// bias for adding temporal dependence*
$h \leftarrow \mathrm{AddNorm}\big(h + \mathrm{CrossAttention}(Q{=}h,\ K{=}m,\ V{=}m;\ \text{noncausal mask},\ B_{\text{read}})\big)$
$h \leftarrow \mathrm{AddNorm}\big(h + \mathrm{TokenFFN}(h)\big)$
$h' \leftarrow h$
// Output decoding (after final layer)
$a \leftarrow \mathrm{ActionHead}(h')$ *// map to action distribution and compute loss*
// Memory track
$B_{\text{rel}} \leftarrow \mathrm{RelativeBias}(p, t)$ *// reversed bias for write*
$u \leftarrow \mathrm{AddNorm}\big(m + \mathrm{CrossAttention}(Q{=}m,\ K{=}h',\ V{=}h';\ \text{noncausal mask},\ B_{\text{write}})\big)$
$\tilde{u} \leftarrow \mathrm{AddNorm}\big(u + \mathrm{MemoryFFN}(u)\big)$
$(m', p') \leftarrow \mathrm{LRU}(m,\ p,\ \tilde{u},\ t)$
return h', (m', p')

3 Method

Many real-world decision-making tasks involve long horizons and partial observability, where key information may appear thousands of steps before it is needed. Standard transformers are limited by a fixed attention window: naive extensions of context length increase cost quadratically, while truncation causes forgetting. Efficient long-term reasoning thus requires a mechanism to store and retrieve task-relevant information across long trajectories. To this end, we propose **ELMUR** (External Layer Memory with Update/Rewrite), a GPT-style [22] transformer decoder augmented with structured external memory. Unlike architectures that simply cache hidden states [23], ELMUR equips each layer with its own memory track and explicit read–write operations, enabling persistent storage and selective updating via the LRU memory management.

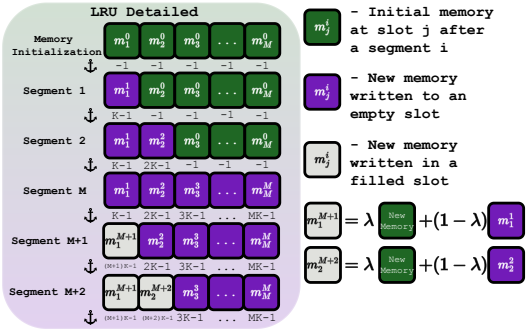


Figure 2: **LRU-based memory management in ELMUR.** Each layer maintains M memory slots, initialized with random vectors (green). As new segments arrive, tokens write updates into empty slots (purple) by full replacement. Once all slots are filled, the least recently used slot is refreshed via a convex update with parameter λ that blends new content with the previous memory (grey). Anchors below each row indicate the timestep of the most recent update. This scheme ensures bounded capacity while preserving long-horizon information.

ELMUR Overview. As shown in Figure 1, each ELMUR layer has two coupled tracks. The **token track** processes observations into actions, while the **memory track** persists across segments. Both interact through cross-attention: memory shapes token representations, and tokens update memory. Interaction occurs via `mem2tok` (read) and `tok2mem` (write) blocks, modulated by relative biases from token timesteps and memory anchors. Trajectories are split into segments, processed sequentially for efficiency and recurrent memory updates. At each segment's end, hidden states update memory, carried forward. LRU memory management fills empty slots first, then refreshes the least used slot by convexly blending old and new information. This bidirectional design provides temporally grounded memory for long-horizon decisions. Algorithm 1 summarizes the method.

Segment-Level Recurrence. Feeding infinitely long sequences into a transformer is infeasible, since self-attention scales quadratically. Splitting into shorter segments reduces cost but complicates information flow. Segment-level recurrence addresses this by treating the transformer as an RNN over segments, passing memory from one segment to the next [23, 24]. In ELMUR, this memory is

realized as layer-local external memory instead of cached activations. Each layer maintains memory that is read within the current segment and updated before moving to the next. Formally, with context length L, a trajectory of length T is partitioned into $S = \lceil T/L \rceil$ segments $\mathcal{S}_i$: $\mathbf{h}^{(i)} = \mathrm{TokenTrack}\big(\mathcal{S}_i, \mathrm{sg}(\mathbf{m}^{i-1})\big)$, where $\mathbf{h}^{(i)} \in \mathbb{R}^{B\times L\times d}$ denotes the hidden states of tokens in segment i, computed from the segment $\mathcal{S}_i$ and the detached memory $\mathrm{sg}(\mathbf{m}^{i-1})$ carried from the previous segment.

Token Track. Within each segment $\mathcal{S}_i$, observations are encoded into token embeddings $\mathbf{x} \in \mathbb{R}^{L\times d}$, where d is the model dimension. The token track models local dependencies and augments them with information from memory $\mathbf{m} \in \mathbb{R}^{M\times d}$. Standard transformers rely on fixed-window self-attention, whereas ELMUR also retrieves information from its external memory via cross-attention, allowing predictions to depend not only on recent tokens but on distant past events stored in memory. Self-attention, equipped with relative positional encodings [23] and a causal mask, models local dependencies within the segment:

$$\mathbf{h}_{\text{sa}} = \mathrm{AddNorm}(\mathbf{x} + \mathrm{SelfAttention}(\mathbf{x}))\,, \tag{1}$$

where $\mathrm{AddNorm}(\cdot)$ denotes a residual connection followed by normalization. Long-term context is handled by external memory. Tokens' hidden states $\mathbf{h}_{\text{sa}}$ then query memory via the `mem2tok`:

$$\mathbf{h}_{\text{mem2tok}} = \mathrm{AddNorm}(\mathbf{h}_{\text{sa}} + \mathrm{CrossAttention}(Q = \mathbf{h}_{\text{sa}},\, K,\, V = \mathbf{m}))\,. \tag{2}$$

Here memory embeddings act as keys and values, with a non-causal mask and a relative bias reflecting token-memory temporal distance. Finally, representations are refined with a feed-forward network (FFN). In contrast to popular Decision Transformer (DT) [25] that employ a standard MLP-based FFN, we adopt a DeepSeek-MoE FFN [26], following the design of DeepSeek-V3 [27]. Mixture-of-Experts (MoE) improve parameter efficiency and specialization by routing tokens to a sparse set of experts, scaling capacity without proportional compute. This design enables expressive updates while keeping inference efficient:

$$\mathbf{h} = \mathrm{AddNorm}(\mathbf{h}_{\text{mem2tok}} + \mathrm{FFN}(\mathbf{h}_{\text{mem2tok}}))\,. \tag{3}$$

The resulting hidden states are then passed to the action head, applied only after the final layer. Training is supervised, minimizing the error between predicted and demonstrated actions, using mean squared loss for continuous spaces and cross-entropy for discrete ones. The loss backpropagates through the entire network to update model parameters.

Memory Track. Reading from memory is not enough for long-horizon reasoning; the model must also write new information. Without an explicit write path, past events would be forgotten or cached inefficiently. The memory track addresses this by allowing tokens to update persistent memory, retaining salient information while overwriting less useful content.

Each layer maintains its own memory embeddings $\mathbf{m} \in \mathbb{R}^{M\times d}$. After processing a segment, token states update memory through the `tok2mem` block:

$$\mathbf{m}_{\text{tok2mem}} = \mathrm{AddNorm}(\mathbf{m} + \mathrm{CrossAttention}(Q = \mathbf{m},\, K, V = \mathbf{h}))\,. \tag{4}$$

As in `mem2tok`, a non-causal mask is applied, but the relative bias is reversed to favor temporally aligned memory embeddings. Updates are then refined by a FFN with residual connection:

$$\mathbf{m}_{\text{new}} = \mathrm{AddNorm}(\mathbf{m}_{\text{tok2mem}} + \mathrm{FFN}(\mathbf{m}_{\text{tok2mem}}))\,, \tag{5}$$

analogous to the token track, the FFN uses a DeepSeek-MoE block instead of a standard MLP.

Finally, $\mathbf{m}_{\text{new}}$ is merged with existing slots via the LRU rule (Figure 2, Algorithm 2), filling empty slots first and otherwise refreshing the least recently used by convex blending. This keeps memory bounded yet consistently updated with relevant information.

Relative Bias. When memory extends across multiple segments, absolute indices become ambiguous: the same token position may correspond to different points in the trajectory. To resolve this, the model requires a signal that encodes relative distances between tokens and memory entries. ELMUR provides this signal through a learned relative bias added to cross-attention logits:

$$\mathrm{Attn}(\mathbf{Q}, \mathbf{K}) = \frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_h}} + \mathbf{B}_{\text{rel}}. \tag{6}$$

The bias $\mathbf{B}_{\text{rel}}$ is derived from pairwise offsets $\Delta = \pm(t-p)$ between a token position t and a memory anchor p (the last update time of a slot). Offsets are clamped to $[-D_{\max}+1, D_{\max}-1]$, where $D_{\max}$ is the maximum relative distance supported by the bias table. These clamped values are shifted into $[0,\ 2D_{\max}-2]$ and used to index a learnable embedding table $\mathbf{E} \in \mathbb{R}^{(2D_{\max}-1)\times H}$, where H is the number of attention heads. Each offset corresponds to a per-head embedding $\mathbf{E}[\Delta] \in \mathbb{R}^H$, and stacking these indices produces

$$\mathbf{B}_{\text{rel}} = \begin{cases} \mathbf{E}[t-p] \in \mathbb{R}^{B\times H\times L\times M}, \texttt{mem2tok (read)} \\ \mathbf{E}[p-t] \in \mathbb{R}^{B\times H\times M\times L}, \texttt{tok2mem (write)}. \end{cases} \tag{7}$$

In the read path (`mem2tok`), the bias prioritizes retrieval from temporally close memory embeddings while keeping distant ones accessible. In the write path (`tok2mem`), offsets are reversed, guiding updates toward memory embeddings aligned with the writing tokens. Both directions draw from the same embedding table $\mathbf{E}$ but can learn distinct patterns. By relying on relative rather than absolute timestep, ELMUR ensures consistent and coherent memory interactions across long horizons.

Memory Management with LRU. External memory must remain bounded: storing every token is infeasible, while naive truncation risks catastrophic forgetting. A principled policy is needed to decide which slots to refresh or preserve as new content arrives. ELMUR employs a Least Recently Used (LRU) block (Figure 2, Algorithm 2) that manages M slots per layer, each holding a vector and an anchor (its last update time). By always updating the least recently used slot, the block ensures bounded capacity while retaining context.

Algorithm 2 LRU update for layer memory. Inputs: current memory (m, p) (may be uninitialized), candidate updates $\tilde{u}$, newest segment time t, blend $\lambda \in [0,1]$, init scale σ. Output: updated memory (m', p').

// Initialization (cold start)
if m, p uninitialized **then**
  $m \leftarrow \mathcal{N}(0, \sigma^2 I)$ *// initial slots*
  $p \leftarrow -\mathbf{1}$ *// sentinel anchors*
end if
// Choose write index
empty $\leftarrow (p < 0)$
if any empty **then**
  $j^\star \leftarrow$ first(empty) *// use first empty slot*
  $\alpha \leftarrow 1$ *// full replacement*
else
  $j^\star \leftarrow \arg\min_j p_j$ *// least recently used*
  $\alpha \leftarrow \lambda$ *// convex blend*
end if
// Integrate
blend $\leftarrow \alpha\, \tilde{u}_{j^\star} + (1-\alpha)\, m_{j^\star}$
$m' \leftarrow m$; $m'_{j^\star} \leftarrow$ blend
$p' \leftarrow p$; $p'_{j^\star} \leftarrow t$
return (m', p')

At training start, **initialization** samples embeddings from $\mathcal{N}(0, \sigma^2 I)$ and marks them empty. While empty slots remain, **full replacement** inserts new vectors directly. Once all slots are filled, the block switches to **convex update**, blending the oldest slot with new content:

$$\mathbf{m}_j^{i+1} = \lambda\, \mathbf{m}_{\text{new}}^{i+1} + (1-\lambda)\, \mathbf{m}_j^i, \tag{8}$$

where $\lambda \in [0,1]$ is a tunable hyperparameter that controls the balance between overwriting and retention. By adjusting λ, one can choose whether memory favors fast plasticity (larger λ) or long-term stability (smaller λ). This policy uses memory capacity fully before overwriting and applies gradual blending thereafter, enabling bounded yet persistent long-horizon memory.

By combining token-level processing with an explicit memory system, ELMUR offers three core advantages: (i) relative-bias cross-attention provides temporally grounded read-write access, (ii) the LRU-based manager ensures bounded capacity while remaining adaptive, and (iii) segment-level recurrence enables scalable learning over long horizons.

4 Theoretical Analysis

Understanding the retention properties of ELMUR's memory is crucial for characterizing its ability to handle long-horizon dependencies. In this section, we analyze how information is preserved or forgotten under the LRU update mechanism. We derive bounds on memory retention and effective horizons, and connect these results to the empirical behaviors observed in long-horizon tasks.

At the core of ELMUR's memory module is the convex update rule with blending factor $\lambda \in [0,1]$. Let fix a memory embedding j at segment index i. If this memory embedding is selected for update

with new content $\mathbf{m}_{\text{new}}^{i+1}$, the rule (Algorithm 2) is $\mathbf{m}_j^{i+1} = \lambda\, \mathbf{m}_{\text{new}}^{i+1} + (1-\lambda)\, \mathbf{m}_j^i$, while all other memory embeddings $n \neq j$ remain unchanged: $\mathbf{m}_n^{i+1} = \mathbf{m}_n^i$. If the memory embedding was empty, the update reduces to full replacement $\mathbf{m}_j^{i+1} = \mathbf{m}_{\text{new}}^{i+1}$.

Proposition 1 (Exponential Forgetting). After k overwrites of memory embedding j, the content evolves as

$$\mathbf{m}_j^{i+k} = (1-\lambda)^k\, \mathbf{m}_j^i + \sum_{u=1}^{k} \lambda(1-\lambda)^{k-u}\, \mathbf{m}_{\text{new}}^{i+u}, \tag{9}$$

where $\mathbf{m}_{\text{new}}^{i+u}$ denotes the write at update $i+u$ (see Appendix A.1 for a full derivation). Consequently, the coefficient of the initial content $\mathbf{m}_j^i$ after k overwrites is $(1-\lambda)^k$, and the contribution of the write performed τ updates earlier is $\lambda(1-\lambda)^{\tau-1}$.

Corollary (Half-life). The number of overwrites $k_{0.5}$ after which the contribution of $\mathbf{m}_j^i$ halves is $k_{0.5} = \frac{\ln(1/2)}{\ln(1-\lambda)} = \frac{\ln 2}{-\ln(1-\lambda)} \sim \frac{\ln 2}{\lambda}$, as $\lambda \to 0$. Thus, smaller λ extends retention, while larger λ accelerates overwriting.

Effective horizon in environment steps. Since only one memory embedding is updated per segment of length L, a memory is overwritten once every M segments in expectation. The *effective retention horizon* $H(\epsilon)$ thus quantifies how many environment steps a stored contribution remains influential before its weight decays below a negligible threshold ϵ, i.e., $H(\epsilon) = M \cdot L \cdot \frac{\ln(\epsilon)}{\ln(1-\lambda)}$. In particular, the half-life in environment steps is $H_{0.5} = M \cdot L \cdot \frac{\ln 2}{-\ln(1-\lambda)} \sim M \cdot L \cdot \frac{\ln 2}{\lambda}$, as $\lambda \to 0$.

Unlike models where all memory is updated at every step (like RNNs), ELMUR's LRU policy ensures (i) memory embeddings not selected for overwrite retain their content exactly until replacement, and (ii) once selected, their contributions decay exponentially with rate λ. This produces a retention horizon that scales linearly with both the number of memory embeddings M and the segment length L, providing a conservative lower bound. In practice, effective horizons are often much longer (Figure 3).

Proposition 2 (Memory Boundedness). A natural question is whether repeated convex updates could cause memory values to grow without limit. We show that, under standard bounded-input assumptions, the norm of every memory embedding remains uniformly bounded throughout training and inference. Suppose that every new write is norm-bounded, $\|\mathbf{m}_{\text{new}}^t\| \leq C$ for some constant $C > 0$, and the initial memory satisfies $\|\mathbf{m}_j^0\| \leq C$. Then for all segments i and slots j, it holds that $\|\mathbf{m}_j^i\| \leq C$. Since each update is a convex combination of the previous and a bounded new values, the memory embedding always remains inside the closed ball of radius C. This guarantees stability of activations even across arbitrarily long trajectories. See Appendix A.2 for the detailed proof.

5 Experiments

We evaluate ELMUR on synthetic [17] tasks, 48 POPGym puzzle/control tasks [18], and robotic manipulation [4], all designed to test memory under partial observability. Our study is guided by the following research questions (RQs):

1. **RQ1**: Does ELMUR retain information across horizons far beyond its attention window?
2. **RQ2**: How well does ELMUR generalize to shorter and longer sequences?
3. **RQ3**: Is ELMUR effective on manipulation tasks with visual observations?
4. **RQ4**: How consistent is ELMUR across puzzles, control, and robotics tasks?
5. **RQ5**: What is the impact of components of ELMUR on its memorization?

5.1 Benchmarks and Baselines

We evaluate ELMUR on three benchmarks designed to isolate memory (Appendix, Figure 7). The **T-Maze** requires recalling an early cue after traversing a long corridor with sparse rewards. The **MIKASA-Robo** suite provides robotic tabletop tasks with RGB observations and continuous actions, including color-recall (`RememberColor`) and delayed reversal (`TakeItBack`). Finally, **POP-Gym** offers a diverse collection of partially observable puzzles and control environments for evaluating general memory use. Detailed descriptions can be found in Appendix A.4.

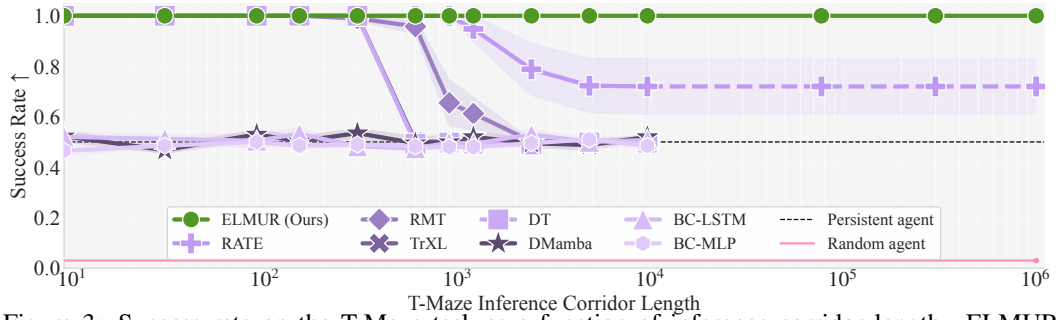


Figure 3: Success rate on the T-Maze task as a function of inference corridor length. ELMUR achieves a **100% success rate** up to corridor lengths of **one million steps**. In this figure, the context length is $L = 10$ with $S = 3$ segments; thus **ELMUR carries information across horizons 100,000 times longer than its context window**.

We compare against baselines spanning sequence models and offline RL for long-horizon tasks. We include transformers — Decision Transformer (DT) [25] and Recurrent Action Transformer with Memory (RATE) [28] — as representative architectures for memory-augmented policy learning. We also evaluate DMamba [29], a state-space model with efficient recurrence, as a recent alternative to attention. For IL/offline RL, we use Behavior Cloning (BC) via MLP as the simplest supervised baseline, Conservative Q-Learning (CQL) [30] as a strong offline RL method, and Diffusion Policy (DP) [31] as a state-of-the-art generative policy. Together, these span transformer, state-space, and offline/generative approaches, providing a competitive reference set for evaluation. We do not compare with online RL baselines, since they assume interactive data collection with exploration, yielding incomparable training budgets. Likewise, we omit real-robot experiments to avoid confounds such as latency, resets, and safety constraints, focusing instead on controlled, reproducible studies.

Experimental Setup. For **RQ1**, we test T-Maze cue retention by training with short contexts (L=10, S=3) and evaluating on corridors up to 10^6 steps. For **RQ2**, we train on 7 T-Maze lengths distributions (9–900) and validate on 11 shorter/longer ones (9–9600) to assess interpolation and extrapolation. For **RQ3**, we use MIKASA-Robo tasks, training by imitation from expert demonstrations and evaluating zero-shot. For **RQ4**, we compare T-Maze, POPGym-48, and MIKASA-Robo to test robustness across synthetic puzzles, control, and robotics. For **RQ5**, we ablate `RememberColor3-v0`, varying M, λ, σ, and (L, S), and remove relative bias, LRU, and per-layer memory to measure component contributions.

Evaluation Protocol. Unless stated otherwise, each model is trained with three (four for T-Maze) independent runs (different initialization). For each run we evaluate on 100 episodes with distinct environment seeds and compute the run mean. We then report the grand mean $\pm$ standard error of the mean (SEM) across the three run means. For per-task leaderboards (e.g., POPGym-48) we apply this protocol per task and aggregate as specified in the benchmark.

Training Details and Hardware. All models are trained from scratch under the same data budgets and preprocessing. We use segment-level recurrence with detached memory between segments; losses are applied on each processed segment. Optimizers, schedulers, and hyperparameters follow the task-specific configuration table in Appendix, Table 5. All experiments were run on a single NVIDIA A100 (80 GB) per job. Training/evaluation code paths, seeds, and environment versions are fixed across methods for reproducibility.

5.2 Results

We evaluate ELMUR on T-Maze, MIKASA-Robo, and POPGym, addressing RQ1–RQ5 on retention, generalization, manipulation, cross-domain robustness, and ablations.

Table 1: Success rates (mean $\pm$ standard error) on MIKASA-Robo tasks, averaged over 3 runs with 100 evaluation seeds. ELMUR outperforms baselines, showing stronger memory in manipulation.

Task	RATE	DT	BC-MLP	CQL-MLP	DP	ELMUR (ours)
`RememberColor3-v0`	0.65±0.04	0.01±0.01	0.27±0.03	0.29±0.01	0.32±0.01	**0.89±0.07**
`RememberColor5-v0`	0.13±0.03	0.07±0.05	0.12±0.01	0.15±0.02	0.10±0.02	**0.19±0.03**
`RememberColor9-v0`	0.09±0.02	0.01±0.01	0.12±0.02	0.15±0.01	0.17±0.01	**0.23±0.02**
`TakeItBack-v0`	0.42±0.24	0.08±0.04	0.33±0.10	0.04±0.01	0.05±0.02	**0.78±0.03**

RQ1: Retention beyond attention. To test memory retention, we train on T-Maze corridors of length T while restricting the context size to $L < T$, forcing the model to solve tasks where the cue must be preserved beyond the native attention span. At validation, we evaluate on much longer corridors — up to one million steps — without increasing L, thereby probing memory retention far beyond the training horizon. ELMUR achieves 100% success even under this extreme extrapolation (Figure 3), implying retention horizons nearly $100{,}000\times$ larger than the attention window ($L{=}10$ with only $S{=}3$ segments used during training).

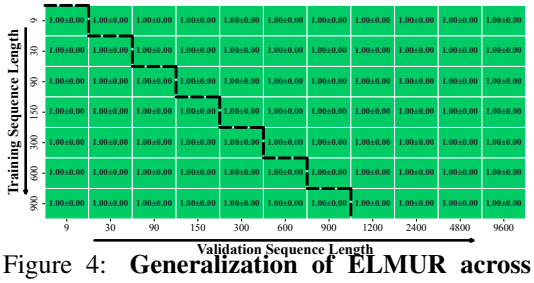


Figure 4: **Generalization of ELMUR across T-Maze lengths.** Each cell shows success rate (mean ± standard error) for training vs. validation lengths. ELMUR transfers perfectly: models trained on shorter sequences retain 100% success up to 9600 steps. Training lengths were split into three equal segments.

RQ2: Generalization across sequence lengths. We train ELMUR on T-Maze with short contexts (3 to 300 steps) and then evaluate across 11 validation lengths ranging from 9 to 9600 steps. The model transfers seamlessly in both directions: it solves tasks shorter than those seen during training without overfitting to a fixed scale, and it also extrapolates to sequences orders of magnitude longer. As shown in Figure 4, ELMUR maintains 100% success across all train/test pairs, demonstrating robust generalization beyond the training horizon.

RQ3: Manipulation with visual observations. Results in Table 1 indicate that ELMUR achieves higher success rates than other baselines on the MIKASA-Robo tasks. In `TakeItBack-v0`, it obtains $0.78{\pm}0.03$ compared to $0.42{\pm}0.24$ for the next-best model, and in `RememberColor[3,5,9]-v0` its performance remains stable as the number of distractors increases. Overall, ELMUR shows more reliable performance under visual interference in manipulation tasks with pixel inputs.

RQ4: Robustness across domains. Across synthetic (T-Maze), control/puzzle (48 POPGym), and robotic (MIKASA-Robo) benchmarks, ELMUR consistently outperforms baselines, generalizing across diverse modalities, actions, and rewards. On POPGym, it achieves the best overall score (10.4), with the largest gains on memory puzzles (1.2 vs. 0.45 for RATE; DT and BC-LSTM score below zero), showing the importance of explicit memory for long-term dependencies. On reactive tasks, ELMUR stays competitive without sacrificing puzzle performance, ranking first on 24 of 48 tasks (full results in Table 4). Figure 5 shows consistent per-task gains over DT, especially on memory-intensive puzzles. Improved retention comes with little overhead: on T-Maze, ELMUR has 2.1M parameters (vs. 1.7M for RATE, 1.8M for DT) yet runs faster per step ($6.8{\pm}0.5$ ms) than RATE ($7.2{\pm}0.3$ ms) and DT ($10.7{\pm}0.1$ ms). Efficiency stems from (i) a short attention window with long-term context handled by bounded memory, so complexity depends on memory size not sequence length, and (ii) MoE feed-forward layers, which raise capacity without proportional compute. Thus, explicit memory is both effective and efficient for long-horizon RL.

Table 2: Aggregated returns on 48 POPGym tasks.

	RATE	DT	Rand.	BC-MLP	BC-LSTM	ELMUR
All (48)	9.5	5.8	-12.2	-6.8	9.0	**10.4**
Puzzle (33)	0.45	-3.5	-14.6	-11.9	-0.2	**1.2**
Reactive (15)	**9.1**	**9.3**	2.3	5.1	**9.1**	**9.2**

RQ5: Ablation Study. We ablate ELMUR's memory design on `RememberColor3-v0` (Figure 6, Table 3). Unless noted, models use *per-layer* memory, relative-bias token–memory cross-attention, and LRU-based updates; *shared memory* denotes embeddings shared across layers. In Figure 6 (b–d)

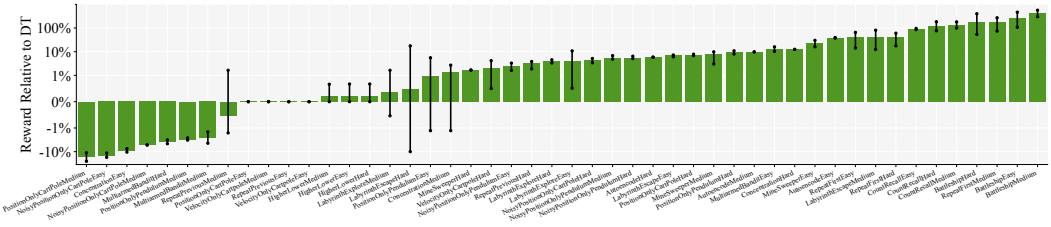


Figure 5: ELMUR compared to DT on all 48 POPGym tasks. Each model was trained with three independent runs, validated over 100 episodes each. Bars show the mean performance with 95% confidence intervals computed over these three means. ELMUR achieves consistent improvements over DT, with the largest gains on memory-intensive puzzles.

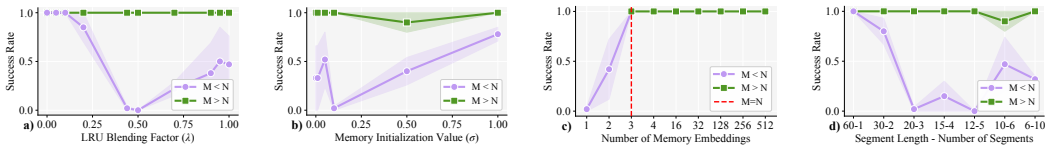


Figure 6: **Ablations of ELMUR's memory hyperparameters on** `RememberColor3-v0`**.** (a) LRU blending factor (λ), (b) memory embeddings initialization (σ), (c) number of memory embeddings (M), and (d) segment configuration ($L-S$). Curves compare settings where the number of memory embeddings is smaller than the required segments to solve the task ($M < N$) versus larger ($M > N$). Results show that sufficient memory capacity ($M \geq N$) yields stable success, while under-provisioned memory ($M < N$) is highly sensitive to λ, σ, and segmentation.

the LRU factor is fixed to $\lambda = 0$ to isolate other effects. Results average three runs of 20 episodes. Performance scales with memory size M: when $M \geq N$ (the number of segments needed), success is near-perfect; when $M < N$, accuracy drops sharply, especially near $M \approx N$ (Figure 6, c–d). Intermediate blending ($\lambda \approx 0.4$–0.6) is unstable (Figure 6, a), while larger initialization σ mitigates collapse (Figure 6, b). Finer recurrence (shorter segments, larger N) stresses capacity unless M scales accordingly. Component ablations confirm that capacity and LRU dominate. Removing LRU leaves stale entries, and removing both LRU and relative bias prevents effective retrieval. Relative bias gives modest gains, while shared memory degrades performance, underscoring the value of layer-local design. Finally, replacing MoE-FFN with MLP-FFN preserves accuracy while improving computational efficiency.

Table 3: Ablation study results.

Setting	Score
Baseline ELMUR	1.00 ± 0.00
Shared memory	0.45 ± 0.03
No rel. bias	0.95 ± 0.05
No LRU	0.43 ± 0.22
No rel. bias; No LRU	0.22 ± 0.11
MoE $\rightarrow$ MLP	1.00 ± 0.00

To confirm that memory mechanisms do not harm performance on fully observable MDPs, we evaluated all models on the simple control task `CartPole-v1` [32]. ELMUR, RATE, RMT, TrXL, BC-MLP, BC-LSTM, and CQL all achieved the maximum return of 500 ± 0, showing that adding memory does not break performance in standard MDP settings.

6 Related Work

Manipulation. Transformer approaches to robotic manipulation can be broadly categorized by their underlying design principles. *Perception-centric visuomotor transformers* focus on multi-view or 3D perception to improve near fully observable control [33, 34]. *Sequence/skill modeling* distills demonstrations into reusable action chunks but remains bottlenecked by limited context [35, 36]. *Planning/value-augmented transformers* integrate transformers with planning or value learning for closed-loop control under finite context [37, 38]. *Alternative backbones* adopt state-space models or diffusion for efficiency, but without persistent memory [39, 31]. *Scaling to VLA* broadens task coverage with language but still suffers from fixed horizons, with some remedies via summarization, feature banks, or hierarchy [40, 16, 1, 3, 7]. ELMUR differs by training as a standard IL transformer while removing the context bottleneck through structured, layer-local external memory.

Memory. Efforts to extend sequence models to long horizons take several forms. *Implicit recurrence and state-space models* compress history in hidden dynamics, offering efficiency but little control over forgetting [41, 42]. *External memory with learned access* provides addressable storage but complicates optimization [43, 44]. *Transformer context extension* retains history via caches or auxiliary slots but keeps memory peripheral [23]. In RL, memory is often implemented through *episodic buffers* for salient events [45] or *sequence-model adaptations* that retrofit transformers for recurrence [21, 46]. Architectures vary in integration: RATE [28] concatenates memory with tokens, Memformer [47] uses global slots, and Block-Recurrent Transformers [48] recycle hidden states. ELMUR instead gives each layer an external memory with dedicated `mem2tok`/`tok2mem` cross-attention and LRU updates, yielding bounded memory for long-horizon tasks (Appendix A.5).

7 Conclusion

We introduced ELMUR, a transformer with layer-local external memory, bidirectional token–memory cross-attention, and an LRU-based update rule. Unlike prior methods, ELMUR integrates explicit memory into every layer, achieving retention horizons up to $100{,}000\times$ beyond the native attention window. Experiments on T-Maze, 48 POPGym tasks, and MIKASA-Robo demonstrate consistent improvements over strong baselines, underscoring reliable credit assignment under partial observability. We envision ELMUR as a simple and extensible framework for long-horizon decision-making with scalable memory in sequential control.

References

[1] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[2] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. pi_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[3] H. Fang, M. Grotz, W. Pumacay, Y. R. Wang, D. Fox, R. Krishna, and J. Duan. Sam2act: Integrating visual foundation model with a memory architecture for robotic manipulation. *arXiv preprint arXiv:2501.18564*, 2025.

[4] E. Cherepanov, N. Kachaev, A. K. Kovalev, and A. I. Panov. Memory, benchmark & robots: A benchmark for solving complex tasks with reinforcement learning. *arXiv preprint arXiv:2502.10550*, 2025.

[5] Z. Ouyang, K. Wang, J. Liu, H. Lu, and W. Zhang. Scil: Stage-conditioned imitation learning for multi-stage manipulation. *IEEE Control Systems Letters*, 2025.

[6] K. Gao, F. Wang, E. Aduh, D. Randle, and J. Shi. Must: Multi-head skill transformer for long-horizon dexterous manipulation with skill progress. *arXiv preprint arXiv:2502.02753*, 2025.

[7] H. Shi, B. Xie, Y. Liu, L. Sun, F. Liu, T. Wang, E. Zhou, H. Fan, X. Zhang, and G. Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2508.19236*, 2025.

[8] R. S. Sutton, A. G. Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[9] K. Zhang, P. Yun, J. Cen, J. Cai, D. Zhu, H. Yuan, C. Zhao, T. Feng, M. Y. Wang, Q. Chen, et al. Generative artificial intelligence in robotic manipulation: A survey. *arXiv preprint arXiv:2503.03464*, 2025.

[10] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[11] Z. Wu, W. Lian, V. Unhelkar, M. Tomizuka, and S. Schaal. Learning dense rewards for contact-rich manipulation tasks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6214–6221. IEEE, 2021.

[12] T. Mu, M. Liu, and H. Su. Drs: Learning reusable dense rewards for multi-stage tasks. *ArXiv*, abs/2404.16779, 2024. URL https://api.semanticscholar.org/CorpusID:269362133.

[13] L. Wang, T. Xu, Y. Lu, and X. Xiao. Reward training wheels: Adaptive auxiliary rewards for robotics reinforcement learning. *arXiv preprint arXiv:2503.15724*, 2025.

[14] M. Zare, P. M. Kebria, A. Khosravi, and S. Nahavandi. A survey of imitation learning: Algorithms, recent developments, and challenges. *IEEE Transactions on Cybernetics*, 2024.

[15] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[16] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[17] T. Ni, M. Ma, B. Eysenbach, and P.-L. Bacon. When do transformers shine in rl? decoupling memory from credit assignment. *Advances in Neural Information Processing Systems*, 36: 50429–50452, 2023.

[18] S. Morad, R. Kortvelesy, M. Bettini, S. Liwicki, and A. Prorok. POPGym: Benchmarking partially observable reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=chDrutUTs0K.

[19] M. Lauri, D. Hsu, and J. Pajarinen. Partially observable markov decision processes in robotics: A survey. *IEEE Transactions on Robotics*, 39(1):21–40, 2022.

[20] M. J. Hausknecht and P. Stone. Deep recurrent q-learning for partially observable mdps. In *AAAI fall symposia*, volume 45, page 141, 2015.

[21] E. Parisotto, F. Song, J. Rae, R. Pascanu, C. Gulcehre, S. Jayakumar, M. Jaderberg, R. L. Kaufman, A. Clark, S. Noury, et al. Stabilizing transformers for reinforcement learning. In *International conference on machine learning*, pages 7487–7498. PMLR, 2020.

[22] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[23] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[24] A. Bulatov, Y. Kuratov, and M. Burtsev. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35:11079–11091, 2022.

[25] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[26] D. Dai, C. Deng, C. Zhao, R. Xu, H. Gao, D. Chen, J. Li, W. Zeng, X. Yu, Y. Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.

[27] A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[28] E. Cherepanov, A. Staroverov, D. Yudin, A. K. Kovalev, and A. I. Panov. Recurrent action transformer with memory. *arXiv preprint arXiv:2306.09459*, 2023.

[29] T. Ota. Decision mamba: Reinforcement learning via sequence modeling with selective state spaces. *arXiv preprint arXiv:2403.19925*, 2024.

[30] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

[31] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[32] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

[33] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.

[34] A. Goyal, V. Blukis, J. Xu, Y. Guo, Y.-W. Chao, and D. Fox. Rvt-2: Learning precise manipulation from few demonstrations. *arXiv preprint arXiv:2406.08545*, 2024.

[35] X. Huang, D. Batra, A. Rai, and A. Szot. Skill transformer: A monolithic policy for mobile manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10852–10862, 2023.

[36] M. Kobayashi, T. Buamanee, Y. Uranishi, and H. Takemura. Ilbit: Imitation learning for robot using position and torque information based on bilateral control with transformer. *IEEJ Journal of Industry Applications*, 14(2):161–168, 2025.

[37] X. Zhang, Y. Liu, H. Chang, L. Schramm, and A. Boularias. Autoregressive action sequence learning for robotic manipulation. *IEEE Robotics and Automation Letters*, 2025.

[38] J. Hu, R. Hendrix, A. Farhadi, A. Kembhavi, R. Martín-Martín, P. Stone, K.-H. Zeng, and K. Ehsani. Flare: Achieving masterful and adaptive robot policies with large-scale reinforcement learning fine-tuning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3617–3624. IEEE, 2025.

[39] J. Liu, M. Liu, Z. Wang, P. An, X. Li, K. Zhou, S. Yang, R. Zhang, Y. Guo, and S. Zhang. Robomamba: Efficient vision-language-action model for robotic reasoning and manipulation. *Advances in Neural Information Processing Systems*, 37:40085–40110, 2024.

[40] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

[41] M. Beck, K. Pöppel, M. Spanring, A. Auer, O. Prudnikova, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter. xlstm: Extended long short-term memory. *Advances in Neural Information Processing Systems*, 37:107547–107603, 2024.

[42] A. Gu and T. Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[43] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwinska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. P. Agapiou, A. P. Badia, K. M. Hermann, Y. Zwols, G. Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu, and D. Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538: 471–476, 2016. URL https://api.semanticscholar.org/CorpusID:205251479.

[44] A. Santoro, S. Bartunov, M. M. Botvinick, D. Wierstra, and T. P. Lillicrap. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, 2016. URL https://api.semanticscholar.org/CorpusID:6466088.

[45] A. Lampinen, S. Chan, A. Banino, and F. Hill. Towards mental time travel: a hierarchical memory for reinforcement learning agents. *Advances in Neural Information Processing Systems*, 34:28182–28195, 2021.

[46] E. Cherepanov, N. Kachaev, A. Zholus, A. K. Kovalev, and A. I. Panov. Unraveling the complexity of memory in rl agents: an approach for classification and evaluation. *arXiv preprint arXiv:2412.06531*, 2024.

[47] Q. Wu, Z. Lan, K. Qian, J. Gu, A. Geramifard, and Z. Yu. Memformer: A memory-augmented transformer for sequence modeling. *arXiv preprint arXiv:2010.06891*, 2020.

[48] D. Hutchins, I. Schlag, Y. Wu, E. Dyer, and B. Neyshabur. Block-recurrent transformers. *Advances in neural information processing systems*, 35:33248–33261, 2022.

[49] Y. Wang, L. Li, B. Chen, A. Li, H. Liu, Q. Zheng, X. Yang, and W. Li. Synthetic pomdps to challenge memory-augmented rl: Memory demand structure modeling. *arXiv preprint arXiv:2508.04282*, 2025.

[50] W. Yue, B. Liu, and P. Stone. Learning memory mechanisms for decision making through demonstrations. *arXiv preprint arXiv:2411.07954*, 2024.

[51] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox. Rvt: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, pages 694–710. PMLR, 2023.

[52] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[53] M. Dalal, A. Mandlekar, C. Garrett, A. Handa, R. Salakhutdinov, and D. Fox. Imitating task and motion planning with visuomotor transformers. *arXiv preprint arXiv:2305.16309*, 2023.

[54] N. Funk, J. Urain, J. Carvalho, V. Prasad, G. Chalvatzaki, and J. Peters. Actionflow: Equivariant, accurate, and efficient policies with spatially symmetric flow matching. *arXiv preprint arXiv:2409.04576*, 2024.

[55] Y. Chebotar, Q. Vuong, K. Hausman, F. Xia, Y. Lu, A. Irpan, A. Kumar, T. Yu, A. Herzog, K. Pertsch, et al. Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. In *Conference on Robot Learning*, pages 3909–3928. PMLR, 2023.

[56] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan. Vima: General robot manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*, 2(3):6, 2022.

[57] X. Li, M. Liu, H. Zhang, C. Yu, J. Xu, H. Wu, C. Cheang, Y. Jing, W. Zhang, H. Liu, et al. Vision-language foundation models as effective robot imitators. *arXiv preprint arXiv:2311.01378*, 2023.

[58] Q. Li, Y. Liang, Z. Wang, L. Luo, X. Chen, M. Liao, F. Wei, Y. Deng, S. Xu, Y. Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

[59] M. Reuss, H. Zhou, M. Rühle, Ö. E. Yağmurlu, F. Otto, and R. Lioutikov. FLOWER: Democratizing generalist robot policies with efficient vision-language-action flow policies. In *7th Robot Learning Workshop: Towards Robots with Human-Level Abilities*, 2025. URL https://openreview.net/forum?id=ifo8oWSLSq.

[60] D. Tarasov, A. Nikulin, I. Zisman, A. Klepach, N. Lyubaykin, A. Polubarov, A. Derevyagin, and V. Kurenkov. Nina: Normalizing flows in action. training vla models with normalizing flows, 2025. URL https://arxiv.org/abs/2508.16845.

[61] H. Zhen, X. Qiu, P. Chen, J. Yang, X. Yan, Y. Du, Y. Hong, and C. Gan. 3d-vla: A 3d vision-language-action generative world model. *arXiv preprint arXiv:2403.09631*, 2024.

[62] Q. Zhao, Y. Lu, M. J. Kim, Z. Fu, Z. Zhang, Y. Wu, Z. Li, Q. Ma, S. Han, C. Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1702–1713, 2025.

[63] R. Zheng, Y. Liang, S. Huang, J. Gao, H. Daumé III, A. Kolobov, F. Huang, and J. Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. *arXiv preprint arXiv:2412.10345*, 2024.

[64] J. Zhang, Y. Guo, X. Chen, Y.-J. Wang, Y. Hu, C. Shi, and J. Chen. Hirt: Enhancing robotic control with hierarchical robot transformers. *arXiv preprint arXiv:2410.05273*, 2024.

[65] B. Han, J. Kim, and J. Jang. A dual process vla: Efficient robotic manipulation leveraging vlm. *arXiv preprint arXiv:2410.15549*, 2024.

[66] J. Liu, H. Chen, P. An, Z. Liu, R. Zhang, C. Gu, X. Li, Z. Guo, S. Chen, M. Liu, C. Hou, M. Zhao, K. alex Zhou, P.-A. Heng, and S. Zhang. Hybridvla: Collaborative diffusion and autoregression in a unified vision-language-action model, 2025. URL https://arxiv.org/abs/2503.10631.

[67] J. Wen, Y. Zhu, J. Li, Z. Tang, C. Shen, and F. Feng. Dexvla: Vision-language model with plug-in diffusion expert for general robot control. *arXiv preprint arXiv:2502.05855*, 2025.

[68] D. Qu, H. Song, Q. Chen, Y. Yao, X. Ye, Y. Ding, Z. Wang, J. Gu, B. Zhao, D. Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

[69] M. J. Kim, C. Finn, and P. Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

[70] J. Wen, Y. Zhu, J. Li, M. Zhu, Z. Tang, K. Wu, Z. Xu, N. Liu, R. Cheng, C. Shen, et al. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 2025.

[71] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, et al. $pi\_\{0.5\}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[72] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[73] G. R. Team, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, T. Armstrong, A. Balakrishna, R. Baruch, M. Bauza, M. Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

[74] C.-Y. Hung, Q. Sun, P. Hong, A. Zadeh, C. Li, U. Tan, N. Majumder, S. Poria, et al. Nora: A small open-sourced generalist vision language action model for embodied tasks. *arXiv preprint arXiv:2504.19854*, 2025.

[75] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997. URL https://api.semanticscholar.org/CorpusID:1915014.

[76] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

[77] B. Peng, E. Alcaide, Q. Anthony, A. Albalak, S. Arcadinho, S. Biderman, H. Cao, X. Cheng, M. Chung, M. Grella, et al. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023.

[78] A. Gu, K. Goel, and C. Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

[79] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[80] J. Weston, S. Chopra, and A. Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[81] A. H. K. Ahmadi. *Memory-based graph networks*. University of Toronto (Canada), 2020.

[82] J. W. Rae, A. Potapenko, S. M. Jayakumar, and T. P. Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

[83] Y. Wu, M. N. Rabe, D. Hutchins, and C. Szegedy. Memorizing transformers. *arXiv preprint arXiv:2203.08913*, 2022.

[84] H. Liu, M. Zaharia, and P. Abbeel. Ring attention with blockwise transformers for near-infinite context. *arXiv preprint arXiv:2310.01889*, 2023.

[85] I. Rodkin, Y. Kuratov, A. Bulatov, and M. Burtsev. Associative recurrent memory transformer. *arXiv preprint arXiv:2407.04841*, 2024.

[86] S. Ding, J. Shang, S. Wang, Y. Sun, H. Tian, H. Wu, and H. Wang. Ernie-doc: A retrospective long-document modeling transformer. *arXiv preprint arXiv:2012.15688*, 2020.

[87] A. Behrouz, P. Zhong, and V. Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024.

[88] E. Parisotto and R. Salakhutdinov. Neural map: Structured memory for deep reinforcement learning. *arXiv preprint arXiv:1702.08360*, 2017.

[89] H. Le, K. Do, D. Nguyen, S. Gupta, and S. Venkatesh. Stable hadamard memory: Revitalizing memory-augmented agents for reinforcement learning. *arXiv preprint arXiv:2410.10132*, 2024.

[90] K. Esslinger, R. Platt, and C. Amato. Deep transformer q-networks for partially observable reinforcement learning. *arXiv preprint arXiv:2206.01078*, 2022.

[91] S. Pramanik, E. Elelimy, M. C. Machado, and A. White. Agalite: Approximate gated linear transformers for online reinforcement learning. *arXiv preprint arXiv:2310.15719*, 2023.

[92] J. Grigsby, J. Sasek, S. Parajuli, D. Adebi, A. Zhang, and Y. Zhu. Amago-2: Breaking the multi-task barrier in meta-reinforcement learning with transformers. *Advances in Neural Information Processing Systems*, 37:87473–87508, 2024.

[93] S. Morad, R. Kortvelesy, S. Liwicki, and A. Prorok. Reinforcement learning with fast and forgetful memory. *Advances in Neural Information Processing Systems*, 36:72008–72029, 2023.

[94] D. Zelezetsky, E. Cherepanov, A. K. Kovalev, and A. I. Panov. Re: Frame–retrieving experience from associative memory. *arXiv preprint arXiv:2508.19344*, 2025.

A Appendix

A.1 Proof of Exponential Forgetting in LRU Updates

Proposition 1 (Exponential Forgetting). Fix a slot j and a segment index i. Suppose this slot is updated k times at segments $i+1, \ldots, i+k$ according to

$$\mathbf{m}_j^t \;=\; \lambda\, \mathbf{m}_{\text{new}}^t \;+\; (1-\lambda)\, \mathbf{m}_j^{t-1}, \qquad t = i+1, \ldots, i+k, \tag{10}$$

with $0 \le \lambda \le 1$. Then

$$\mathbf{m}_j^{i+k} \;=\; (1-\lambda)^k\, \mathbf{m}_j^i \;+\; \sum_{u=1}^{k} \lambda(1-\lambda)^{k-u}\, \mathbf{m}_{\text{new}}^{i+u}. \tag{11}$$

Consequently, the coefficient of the initial content $\mathbf{m}_j^i$ after k overwrites is $(1-\lambda)^k$, and the coefficient of the write performed τ updates ago (at segment $i+\tau$) is $\lambda(1-\lambda)^{\tau-1}$.

Proof. We prove Equation 11 by induction on k.

Base case $k = 1$. By the update rule,

$$\mathbf{m}_j^{i+1} \;=\; \lambda\, \mathbf{m}_{\text{new}}^{i+1} + (1-\lambda)\, \mathbf{m}_j^{i}, \tag{12}$$

which matches Equation 11 with $k = 1$.

Induction step. Assume Equation 11 holds for some $k \geq 1$. For $k+1$,

$$\mathbf{m}_j^{i+k+1} = \lambda\, \mathbf{m}_{\text{new}}^{i+k+1} + (1-\lambda)\, \mathbf{m}_j^{i+k}. \tag{13}$$

Insert the induction hypothesis for $\mathbf{m}_j^{i+k}$:

$$\mathbf{m}_j^{i+k+1} = \lambda\, \mathbf{m}_{\text{new}}^{i+k+1} + (1-\lambda)\left[(1-\lambda)^k\, \mathbf{m}_j^{i} + \sum_{u=1}^{k} \lambda(1-\lambda)^{k-u}\, \mathbf{m}_{\text{new}}^{i+u}\right]. \tag{14}$$

Distribute $(1-\lambda)$ and regroup terms:

$$\mathbf{m}_j^{i+k+1} = (1-\lambda)^{k+1}\mathbf{m}_j^{i} + \sum_{u=1}^{k} \lambda(1-\lambda)^{(k+1)-u}\, \mathbf{m}_{\text{new}}^{i+u} + \lambda\, \mathbf{m}_{\text{new}}^{i+k+1}, \tag{15}$$

which is exactly Equation 11 with k replaced by $k+1$. This completes the induction.

Finally, the coefficients in Equation 11 form a convex combination:

$$(1-\lambda)^k \;+\; \sum_{u=1}^{k} \lambda(1-\lambda)^{k-u} = (1-\lambda)^k + \lambda \sum_{r=0}^{k-1} (1-\lambda)^r = 1, \tag{16}$$

so it is meaningful to call them "fractions" of contribution. ∎

Corollary (Half-life). The number of overwrites $k_{0.5}$ after which the contribution of $\mathbf{m}_j^i$ halves satisfies

$$(1-\lambda)^{k_{0.5}} = \tfrac{1}{2} \;\;\Longrightarrow\;\; k_{0.5} = \frac{\ln(1/2)}{\ln(1-\lambda)}. \tag{17}$$

Equivalently,

$$k_{0.5} = \frac{\ln 2}{-\ln(1-\lambda)}. \tag{18}$$

Using the Maclaurin series expansion $\ln(1-\lambda) \sim -\lambda$ as $\lambda \to 0$, we obtain

$$\lim_{\lambda \to 0} \; k_{0.5} \cdot \lambda \;=\; \ln 2. \tag{19}$$

Hence,

$$k_{0.5} \;\sim\; \frac{\ln 2}{\lambda} \quad \text{as } \lambda \to 0, \tag{20}$$

showing that smaller λ yields longer retention horizons, while larger λ overwrites past content more aggressively.

A.2 Boundedness of Memory Embeddings

Fix a slot j and consider its update rule,

$$\mathbf{m}_j^{i+1} = \lambda\, \mathbf{m}_{\text{new}}^{i+1} + (1-\lambda)\, \mathbf{m}_j^{i}, \qquad 0 \leq \lambda \leq 1. \tag{21}$$

We prove the claim by induction on i.

Base case. At $i = 0$, the assumption gives $\|\mathbf{m}_j^0\| \leq C$.

Induction step. Assume $\|\mathbf{m}_j^i\| \leq C$ for some $i \geq 0$. Using the update rule,

$$\|\mathbf{m}_j^{i+1}\| = \|\lambda\, \mathbf{m}_{\text{new}}^{i+1} + (1-\lambda)\, \mathbf{m}_j^{i}\|. \tag{22}$$

By the triangle inequality and the inductive hypothesis,

$$\|\mathbf{m}_j^{i+1}\| \leq \lambda\|\mathbf{m}_{\text{new}}^{i+1}\| + (1-\lambda)\|\mathbf{m}_j^{i}\| \leq \lambda C + (1-\lambda)C = C. \tag{23}$$

Thus $\|\mathbf{m}_j^{i+1}\| \leq C$, completing the induction.

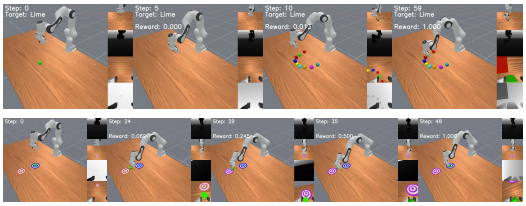
Therefore, for all i, the memory embedding satisfies $\|\mathbf{m}_j^i\| \leq C$. ∎

A.3 Memory-intensive environments

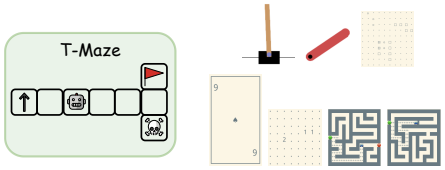
Event–recall pairs and correlation horizon. Following [46], let $\alpha_{t_e}^{\Delta t}$ denote an event of duration Δt starting at time t_e, and let $\beta_{t_r}(\alpha_{t_e}^{\Delta t})$ be a later decision point that must recall information from that event. Define the **correlation horizon** as

$$\xi \;=\; t_r - t_e - \Delta t + 1. \tag{24}$$

For an environment, collect all event–recall horizons into the set $\Xi = \{\xi_n\}_n$.



(a) MIKASA-Robo: `RememberColor9-v0` (top) and `TakeItBack-v0` (bottom).



(b) T-Maze.

(c) Some POPGym tasks.

Figure 7: **Environments used in experiments.**

Recent works have proposed alternative but complementary definitions of memory-intensive environments. For instance, Wang et al. [49] formalize **memory demand structures** that capture the minimal past sufficient to predict future transitions and rewards, while Yue et al. [50] introduce **memory dependency pairs** to annotate which past observations must be recalled for correct decisions. Both perspectives emphasize controllable ways to scale memory difficulty, either by increasing order/span or by manipulating dependency graphs. In this paper, we adopt the event–recall horizon framework for clarity and analytical tractability, but note that these alternative views are broadly consistent and provide useful tools for designing and benchmarking memory-intensive tasks.

Definition (memory-intensive environment). A POMDP $\tilde{M}_P$ is **memory-intensive** if $\min_n \Xi > 1$; i.e., every relevant decision depends on information separated by at least one intervening step, making reactive (myopic) policies insufficient. This definition cleanly separates POMDPs that genuinely require memory from those that are effectively MDP-like.

A.4 Memory-Intensive Environments

The memory-intensive environments used in this work are presented in Figure 7.

T-Maze. The T-Maze [17] features a corridor ending in a junction with two goals. At the start, one goal is randomly revealed, and the agent must recall this cue after traversing the corridor to choose the correct branch. Observations are vectors; actions are discrete. Rewards are sparse, provided only upon reaching the correct goal. The task tests whether the model can retain early cues across long delays.

MIKASA-Robo benchmark. We further evaluate on the MIKASA-Robo benchmark [4], which provides robotic tabletop manipulation tasks designed for memory evaluation. Each environment simulates a 7-DoF arm with a two-finger gripper. Observations are paired RGB images ($3 \times 128 \times 128$) from a static and wrist camera; actions are continuous (7 joints + gripper). Rewards are binary, given only on task success. We study two families of MIKASA-Robo tasks: (i) `RememberColor[3,5,9]-v0`, where the agent must recall the color of a hidden cube after a delay with distractors, and (ii) `TakeItBack-v0`, where the agent first moves a cube to a goal, then must return it once the goal changes.

POPGym benchmark. POPGym benchmark [18] is a large suite of 48 partially observable environments designed to stress agent memory. The tasks span two categories: (i) diagnostic memory puzzles, which require agents to remember cues or solve algorithmic sequence problems (e.g., copy, reverse, n-back, and long-horizon T-Mazes), and (ii) partially observable control tasks, which

adapt classic control benchmarks such as CartPole, MountainCar, and LunarLander to observation-limited settings. This diversity allows POPGym to test both symbolic memory skills and generalization in continuous control, providing a broad and challenging benchmark for evaluating memory-augmented RL methods. Detailed results across all 48 POPGym tasks for each of considered baselines are presented in Table 4.

A.5 Extended Related Work

Transformers for manipulation. Work on transformer-based manipulation splits into characteristic families with complementary strengths and limits. *Perception-centric visuomotor transformers* emphasize 3D or multi-view geometry to map pixels to precise end-effector actions under (near) full observability, excelling at spatial alignment but assuming short temporal credit assignment: PerAct [33], RVT [51], RVT-2 [34]. *Sequence/skill models* distill demonstrations into reusable action chunks and long-horizon trajectories, improving sample efficiency but remaining bottlenecked by finite context windows: ACT [52], Skill Transformer [35], ILBiT [36]. *Planning/RL-driven transformers* blend sequence modeling with value learning or planning for closed-loop control under uncertainty, yet still inherit fixed-context limitations: OPTIMUS [53], ActionFlow [54], Q-Transformer [55], CCT/ARP [37], FLaRe [38]. *Alternative backbones* (state-space, diffusion) target long continuous horizons and smooth control but lack explicit persistent state: RoboMamba [39], Diffusion Policy [31].

VLA models for manipulation. VLA models broaden task coverage by conditioning on language and scaling data, while keeping the transformer core unchanged and context-bounded. *Instruction-following generalists* demonstrate broad skill repertoires with language prompts but no explicit long-term memory: RT-1 [15], RT-2 [40], Octo [16], OpenVLA [1], VIMA [56]. *Efficiency/modularity variants* freeze large encoders and train lightweight adapters or experts for practicality at scale: RoboFlamingo [57], CogACT [58], FLOWER [59], NinA [60]. *Reasoning/hierarchy extensions* inject geometric priors, step-wise reasoning, or multi-level control while still relying on finite windows: 3D-VLA [61], CoT-VLA [62], TraceVLA [63], HiRT [64], DP-VLA [65]. *Specialized and hybrid designs* tailor the backbone to domain constraints (dexterous hands, spatial priors) or mix diffusion with autoregression: HybridVLA [66], DexVLA [67], SpatialVLA [68], OpenVLA-OFT [69]. *Generalist robot agents* pursue open-world embodiment with growing breadth but inherit the same temporal limitations: TinyVLA [70], π_0 [2], $\pi_{0.5}$ [71], GR00T N1 [72], Gemini Robotics [73], NORA [74].

Memory in Deep Learning. Mechanisms for long-term information fall into three broad tracks. *Implicit recurrence and long-range sequence models* retain information in hidden dynamics but offer limited, indirect control over storage and forgetting: LSTM [75], xLSTM [41], linear-attention Transformers [76], RWKV [77], S4 [78], Mamba [42]. *Explicit external memory with learned read–write* provides addressable storage with content-based access, trading off simplicity for optimization/scaling complexity: NTM [79], DNC [43], Memory Networks [80], differentiable memory for meta-learning [44], recent variants [81]. *Transformer context extension* pushes horizons via cached activations, compression, or auxiliary memory modules but typically keeps memory peripheral to the core token computation: Transformer-XL [23], Compressive Transformer [82], Memorizing Transformer [83], Ring Attention [84], Memformer [47], associative-memory Transformers [85], ERNIE-style memory [86], test-time memorization (TiTANS) [87].

Memory in RL. In partially observed decision processes, memory is not optional; it is the state estimator. *Spatial/episodic buffers* externalize salient facts in read–write maps or associative stores (navigational and episodic use-cases): Neural Map [88], HCAM [45], Stable Hadamard Memory [89]. *Sequence-model adaptations* retrofit transformers for recurrence and stability in RL, improving long-horizon training but leaving persistence bounded by context: DTQN [90], GTrXL [21], AGaLiTe [91], AMAGO-2 [92]. *Evaluation frameworks* formalize memory demands and failure modes under partial observability: [46, 50, 49]. *Transformers with external stores for RL* insert ex-

Table 4: **POPGym benchmark results.** Reported scores are mean ± standard error, averaged over 3 runs and evaluated with 100 random seeds. Across 21 partially observable tasks spanning puzzle and control domains, ELMUR achieves the best performance on 12 tasks, underscoring the effectiveness of its memory module for reasoning under partial observability.

POPGym-48 Task	RATE	DT	Random	BC-MLP	BC-LSTM	ELMUR	Expert PPO-GRU
AutoencodeEasy-v0	-0.29±0.00	-0.47±0.00	-0.50±0.00	-0.47±0.00	-0.32±0.00	-0.26±0.00	-0.26
AutoencodeMedium-v0	-0.46±0.00	-0.49±0.00	-0.50±0.01	-0.50±0.00	-0.44±0.00	-0.44±0.00	-0.43
AutoencodeHard-v0	-0.47±0.00	-0.49±0.00	-0.50±0.00	-0.49±0.00	-0.47±0.00	-0.46±0.00	-0.48
BattleshipEasy-v0	-0.81±0.02	-0.93±0.03	-0.46±0.01	-1.00±0.00	-0.49±0.01	-0.79±0.02	-0.35
BattleshipMedium-v0	-0.92±0.01	-0.97±0.01	-0.41±0.00	-1.00±0.00	-0.67±0.01	-0.79±0.01	-0.40
BattleshipHard-v0	-0.91±0.02	-0.91±0.03	-0.39±0.01	-1.00±0.00	-0.81±0.02	-0.80±0.00	-0.43
ConcentrationEasy-v0	-0.06±0.02	-0.05±0.01	-0.19±0.01	-0.92±0.00	-0.14±0.00	-0.14±0.01	-0.12
ConcentrationMedium-v0	-0.25±0.00	-0.25±0.01	-0.19±0.00	-0.92±0.00	-0.19±0.01	-0.24±0.00	-0.44
ConcentrationHard-v0	-0.84±0.00	-0.84±0.00	-0.84±0.00	-0.88±0.00	-0.84±0.00	-0.82±0.00	-0.87
CountRecallEasy-v0	0.07±0.01	-0.46±0.01	-0.93±0.00	-0.92±0.00	0.05±0.00	0.03±0.01	0.22
CountRecallMedium-v0	-0.54±0.00	-0.81±0.02	-0.93±0.00	-0.92±0.00	-0.56±0.00	-0.56±0.01	-0.55
CountRecallHard-v0	-0.47±0.01	-0.75±0.03	-0.88±0.00	-0.88±0.00	-0.47±0.00	-0.47±0.00	-0.48
HigherLowerEasy-v0	0.50±0.00	0.50±0.00	0.00±0.01	0.47±0.00	0.50±0.00	0.50±0.00	0.51
HigherLowerMedium-v0	0.52±0.00	0.51±0.00	0.01±0.01	0.50±0.00	0.51±0.01	0.51±0.00	0.49
HigherLowerHard-v0	0.50±0.00	0.50±0.00	-0.01±0.00	0.49±0.00	0.50±0.00	0.50±0.00	0.49
LabyrinthEscapeEasy-v0	0.95±0.00	0.80±0.01	-0.39±0.00	0.72±0.05	0.92±0.01	0.92±0.00	0.95
LabyrinthEscapeMedium-v0	-0.56±0.01	-0.67±0.04	-0.84±0.04	-0.71±0.03	-0.69±0.02	-0.54±0.01	-0.49
LabyrinthEscapeHard-v0	-0.81±0.01	-0.82±0.01	-0.94±0.01	-0.89±0.01	-0.86±0.00	-0.82±0.01	-0.94
LabyrinthExploreEasy-v0	0.95±0.00	0.88±0.06	-0.34±0.01	0.87±0.01	0.93±0.00	0.96±0.00	0.96
LabyrinthExploreMedium-v0	0.88±0.00	0.86±0.01	-0.61±0.00	0.45±0.01	0.82±0.01	0.86±0.01	0.87
LabyrinthExploreHard-v0	0.79±0.00	0.77±0.01	-0.73±0.00	0.26±0.01	0.71±0.01	0.84±0.00	0.79
MineSweeperEasy-v0	0.15±0.03	-0.33±0.04	-0.26±0.03	-0.47±0.01	0.20±0.00	-0.17±0.02	0.28
MineSweeperMedium-v0	-0.20±0.00	-0.37±0.02	-0.39±0.01	-0.48±0.00	-0.16±0.00	-0.32±0.00	-0.10
MineSweeperHard-v0	-0.44±0.00	-0.40±0.01	-0.43±0.00	-0.49±0.00	-0.35±0.01	-0.39±0.01	-0.27
MultiarmedBanditEasy-v0	0.37±0.01	0.27±0.01	0.02±0.00	0.05±0.00	0.17±0.02	0.43±0.01	0.62
MultiarmedBanditMedium-v0	0.32±0.01	0.35±0.01	0.01±0.00	0.21±0.01	0.14±0.00	0.32±0.01	0.59
MultiarmedBanditHard-v0	0.22±0.03	0.27±0.01	0.01±0.00	0.01±0.00	0.17±0.01	0.22±0.01	0.43
NoisyPositionOnlyCartPoleEasy-v0	0.88±0.03	0.87±0.02	0.11±0.00	0.23±0.00	0.44±0.01	0.59±0.02	0.98
NoisyPositionOnlyCartPoleMedium-v0	0.33±0.01	0.34±0.00	0.12±0.01	0.18±0.00	0.25±0.01	0.27±0.00	0.57
NoisyPositionOnlyCartPoleHard-v0	0.18±0.01	0.17±0.01	0.11±0.00	0.16±0.00	0.22±0.01	0.22±0.01	0.36
NoisyPositionOnlyPendulumEasy-v0	0.87±0.00	0.84±0.01	0.27±0.01	0.31±0.00	0.88±0.00	0.88±0.00	0.90
NoisyPositionOnlyPendulumMedium-v0	0.68±0.00	0.63±0.01	0.27±0.01	0.30±0.00	0.72±0.00	0.72±0.00	0.73
NoisyPositionOnlyPendulumHard-v0	0.60±0.01	0.56±0.01	0.26±0.00	0.28±0.00	0.66±0.00	0.65±0.00	0.67
PositionOnlyCartPoleEasy-v0	0.93±0.03	1.00±0.00	0.12±0.00	0.15±0.00	0.17±0.00	1.00±0.00	1.00
PositionOnlyCartPoleMedium-v0	0.07±0.00	0.34±0.08	0.05±0.00	0.09±0.00	0.12±0.00	0.11±0.00	1.00
PositionOnlyCartPoleHard-v0	0.05±0.01	0.03±0.00	0.04±0.00	0.05±0.00	0.06±0.00	0.10±0.00	1.00
PositionOnlyPendulumEasy-v0	0.54±0.02	0.51±0.03	0.27±0.00	0.29±0.00	0.91±0.00	0.52±0.00	0.92
PositionOnlyPendulumMedium-v0	0.49±0.01	0.55±0.01	0.26±0.00	0.30±0.00	0.89±0.00	0.50±0.01	0.88
PositionOnlyPendulumHard-v0	0.47±0.01	0.49±0.01	0.26±0.00	0.28±0.00	0.82±0.00	0.63±0.01	0.82
RepeatFirstEasy-v0	1.00±0.00	0.45±0.16	-0.49±0.01	-0.50±0.00	1.00±0.00	1.00±0.00	1.00
RepeatFirstMedium-v0	0.99±0.01	-0.21±0.18	-0.50±0.00	-0.50±0.00	0.99±0.01	0.99±0.00	1.00
RepeatFirstHard-v0	0.10±0.02	0.42±0.14	-0.50±0.00	-0.50±0.00	-0.50±0.00	0.99±0.01	0.99
RepeatPreviousEasy-v0	1.00±0.00	1.00±0.00	-0.49±0.01	-0.52±0.00	1.00±0.00	1.00±0.00	1.00
RepeatPreviousMedium-v0	-0.38±0.01	-0.38±0.00	-0.50±0.01	-0.50±0.00	-0.38±0.00	-0.39±0.00	-0.39
RepeatPreviousHard-v0	-0.46±0.00	-0.47±0.00	-0.51±0.00	-0.48±0.00	-0.45±0.00	-0.46±0.00	-0.48
VelocityOnlyCartpoleEasy-v0	1.00±0.00	1.00±0.00	0.11±0.00	0.99±0.00	1.00±0.00	1.00±0.00	1.00
VelocityOnlyCartpoleMedium-v0	1.00±0.00	1.00±0.00	0.06±0.00	0.83±0.01	1.00±0.00	1.00±0.00	1.00
VelocityOnlyCartpoleHard-v0	1.00±0.00	0.96±0.02	0.04±0.00	0.63±0.00	1.00±0.00	1.00±0.00	0.99
Sum of returns	9.54	5.80	-12.24	-6.83	8.96	**10.41**	16.51

plicit memory alongside the policy, but often as a sequence-level attachment: RATE [28], FFM [93], Re:Frame [94].

Memory in manipulation tasks. Long-horizon, partially observed manipulation has motivated three practical extensions. *Trajectory summarization* compresses the past into a short token set, trading completeness for compactness: TraceVLA [63]. *External feature banks* cache recent visual features for retrieval, focusing attention but leaving distant history underrepresented: SAM2Act+ [3]. *Structured memory modules* interface an explicit store with the policy to persist scene/task variables: MemoryVLA [7]. *Hierarchies* shift temporal burden to slow planners or high-level controllers, which may mask but not remove the need for persistent state at the policy layer: HiRT [64], DP-VLA [65]. For evaluation, *targeted benchmarks* isolate memory factors: MemoryBench [3]; broader suites capture multiple memory types and delays: MIKASA-Robo [4].

Parameter	RememberColor	TakeItBack	T-Maze	POPGym-AutoencodeEasy	CartPole-v1
d_{model}	128	32	128	64	128
Routed d_{ff}	128	256	32	128	128
Shared d_{ff}	128	32	512	256	256
Layers	4	2	2	12	4
Heads	16	16	2	4	4
Experts (MoE)	16	1	2	1	4
Shared Experts	1	1	2	2	1
Top-k routing	2	1	3	1	2
Memory size	256	32	2	8	16
Memory init std	0.1	0.001	0.001	0	0.01
Memory dropout	0.06	0.23	0.01	0.17	0.05
LRU blend α	0.40	0.20	0.05	0.80	0.9
Dropout	0.13	0.01	0.10	0.14	0.10
Dropatt	0.30	0.18	0.17	0.26	0.10
Label smoothing	0.21	0.20	0.16	0.22	0.00
Context length	20	60	10	35	30
Batch size	64	64	128	128	512
Learning rate	2.05e-4	2.57e-4	2.06e-4	1.16e-4	3.0e-4
Warmup steps	30000	30000	10000	50000	1000
Cosine decay	True	False	True	False	True
LR end factor	0.1	0.01	1.0	0.01	0.1
Weight decay	0.001	0.01	0.0001	0.1	0.01
Epochs	200	300	1000	800	100
Grad clip	5	1	5	5	1.0
Beta1	0.99	0.9	0.95	0.99	0.9
Beta2	0.99	0.99	0.999	0.99	0.999

Table 5: **Hyperparameters for ELMUR.** We report all architecture and training parameters used across tasks.

A.6 Training Process

When training ELMUR, we compute the loss on every segment processed recursively, detaching the memory state between segments (i.e., without backpropagation through time). The choice of loss depends on the task domain: for T-Maze, CartPole-v1, and a subset of POPGym tasks with discrete actions, we apply a cross-entropy loss; for MIKASA-Robo and the remaining POPGym tasks with continuous actions, we use a mean-squared error (MSE) loss. The ELMUR hyperparameters used in the experiments are presented in The ELMUR hyperparameters used in the experiments are presented in Table 5.

For data, we follow a consistent offline imitation-learning setup. Each MIKASA-Robo environment provides a dataset of 1000 expert demonstrations generated by a PPO policy trained with oracle-level state access. For T-Maze, we collect 6000 successful oracle-level trajectories. For POPGym, we adopt the datasets of Morad et al. [18], consisting of 3000 trajectories per environment generated by a PPO-GRU expert. Finally, for CartPole-v1, we use 1000 successful trajectories collected from a pre-trained PPO policy.

A.7 Limitations

ELMUR uses a simple LRU rule with fixed blending, which makes its memory mechanism transparent and easy to analyze, though future work could explore adaptive variants. Segment-level recurrence adds a small cross-attention cost per layer, but this cost scales with the fixed number of memory slots rather than sequence length, making efficiency predictable even in long horizons. MoE-based FFNs already provide parameter efficiency at our scale, and larger architectures may further amplify this benefit. Finally, our study focuses on synthetic, POPGym, and simulated robotic tasks under IL, giving controlled and reproducible insights; extending to online RL and real-robot deployments offers promising next steps.