MATE: SOLVING CONTEXTUAL MARKOV DECISION PROCESSES WITH MEMORY OF ACCUMULATED TRANSITION EMBEDDINGS

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose MATE, a simple yet effective memory architecture for Contextual Markov Decision Processes (CMDPs), a family of MDPs parameterized by an unobserved context. In CMDPs, an optimal agent adapts online by updating its posterior belief over contexts with every new transition. MATE replaces this intractable update with a memory that additively accumulates transition embeddings, leveraging the posterior's permutation invariance to retain provably sufficient expressiveness. This design avoids both the growing per-step rollout cost of Transformers and the backpropagation through time of recurrent networks. We further propose STORE, a training algorithm that exploits the additive structure of MATE to train on a random subset of transitions per episode while reusing stored embeddings for the rest. STORE keeps the low cost of truncated-window training, yet it places no limit on the memory horizon and matches full-episode training in expectation. Across diverse benchmarks, MATE matches the performance of standard sequence-model baselines while being substantially faster to compute. Replacing truncated-window training with STORE lets MATE learn dependencies far longer than the subset size and yields further performance gains.

1 INTRODUCTION

Contextual Markov Decision Processes (CMDPs) Hallak et al. (2015) extend the standard MDP by letting the environment vary with an unknown context. Such variation is unavoidable in the real world, so solving CMDPs enables RL agents to move from fixed simulation environments to practical applications such as sim-to-real transfer, healthcare, and autonomous driving Kirk et al. (2023). To solve a CMDP, an agent must infer the unknown context from its interaction history. A memoryless policy in standard RL acts on the current state alone and cannot use this history.

A prominent solution is memory-based RL, where sequence models are trained end-to-end to encode transition histories into a policy-accessible memory. Ni et al. (2022) demonstrated the efficacy of this approach using RNNs, outperforming prior specialized methods. More recently, Grigsby et al. (2024a;b) achieved strong performance in multi-task and long-horizon domains by leveraging Transformer architectures. However, these architectures face distinct limitations: RNNs suffer from gradient instabilities Bengio et al. (1994); Pascanu et al. (2013) and preclude temporal parallelization during the update step Lu et al. (2023). Conversely, Transformers incur quadratic computational costs with respect to sequence length during both the update step and trajectory rollout Beck et al. (2025). Both are therefore usually trained on a short contiguous subset of each episode (Williams & Peng, 1990; Ni et al., 2022; Grigsby et al., 2024b). However, gradient does not flow beyond the subset, so the memory horizon is limited to the subset size (Morad et al., 2024). Training on full episodes removes this limit, but the update cost then grows with the episode length.

To address these limitations, we propose **MATE** (**M**emory of **A**ccumulated **T**ransition **E**mbeddings), a memory architecture inspired by a structural property of CMDPs (Figure 1). Since transitions are conditionally independent given the context, the posterior over the context is invariant to the order of transitions, so an optimal agent needs only an order-invariant summary of its history. MATE therefore represents the memory as a sum of transition embeddings (Zaheer et al., 2017), with no recurrent layers or attention (see lower-left of Figure 1). Despite this simplicity, we prove that an

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.

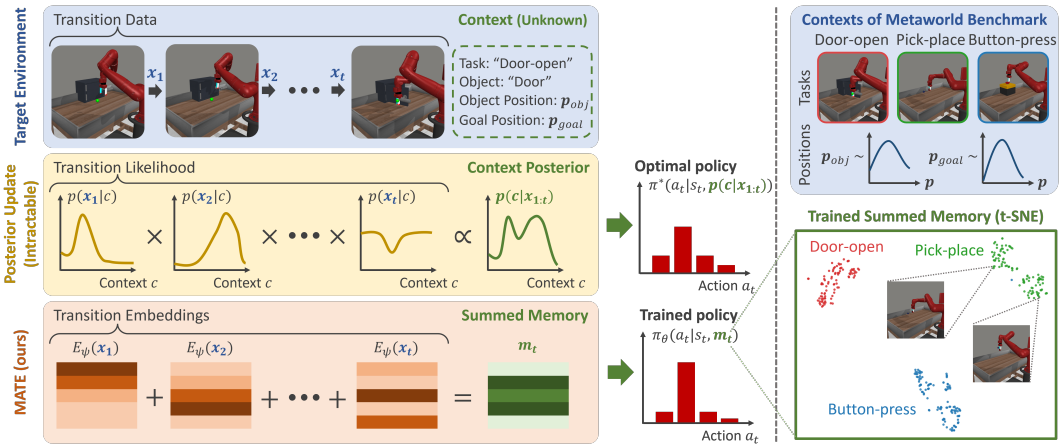


Figure 1: **Overview of MATE.** MATE represents the memory m_t as a summation of transition embeddings, serving as a tractable substitute for the intractable posterior $p(c|x_{1:t})$. By preserving the permutation invariance of the posterior, π_θ is provably capable of representing the optimal policy π^*, despite its structural simplicity.

agent equipped with MATE can still represent the optimal policy. The sum is also cheap: each rollout step updates it in constant time like an RNN, and each update step computes it in parallel across the sequence like a Transformer, at a cost linear in the sequence length.

The purely additive memory of MATE further enables **STORE** (**S**ubset **T**raining **O**ver **R**eused **E**mbeddings), a training algorithm that resolves the trade-off between update cost and memory horizon (Algorithm 1). The embedding of every transition is stored at rollout. Each update then re-embeds a random subset of the transitions in an episode and keeps the stored embeddings for the others, so the memory at every step is still computed from the whole history while the update costs no more than subset training. Since the subset is drawn at random rather than as a contiguous block, gradient reaches transitions from any earlier point in the episode, so the memory horizon is no longer limited by the subset size. We also prove that, if the stored embeddings are up to date, the update equals the full-episode gradient in expectation.

We evaluate MATE on MuJoCo, Meta-World, and T-Maze, chosen to cover adaptation to reward, dynamics, and task variation as well as long-term memory and credit assignment. We compare against GPT-2 (Radford et al., 2019), LSTM (Hochreiter & Schmidhuber, 1997), SplAgger (Beck et al., 2024), and Mamba (Gu & Dao, 2024). MATE trains in the least time on every task, and the gap grows with the episode length. At the same time, it performs on par with the four baselines, and no single model is best on every task. Training MATE with STORE extends its memory horizon well beyond the subset size on T-Maze and also improves its return on MuJoCo and Meta-World. Further experiments show that MATE is compatible with an auxiliary contrastive loss and that it works from pixel observations.

2 Related Work

2.1 Meta-Reinforcement Learning

Meta-RL studies agents that adapt to new tasks from interaction history (Beck et al., 2025). *Task-inference* methods estimate a latent task variable with dedicated objectives (Rakelly et al., 2019; Zintgraf et al., 2020; Imagawa et al., 2022; Li et al., 2024), whereas *black-box* methods compress the history into memory and train the whole agent with the RL objective (Duan et al., 2016; Ni et al., 2022; Beck et al., 2023; Grigsby et al., 2024a;b). MATE is a black-box method, but its memory rests on the order invariance of the CMDP context posterior instead of an order-sensitive sequence model.

2.2 Memory Architectures for RL

Memory in RL is typically a recurrent network, which runs at constant per-step cost during rollout but is hard to optimize over long horizons and cannot be parallelized over time (Bengio et al., 1994;

Pascanu et al., 2013; Lu et al., 2023), or a Transformer, which improves long-range memory at a cost that grows with the history length (Ni et al., 2023; Grigsby et al., 2024a; Beck et al., 2025). State-space models, Mamba, Fast and Forgetful Memory, and memoroids improve this trade-off (Lu et al., 2023; Gu & Dao, 2024; Morad et al., 2023; 2024), but all of them are still sequence models for ordered inputs, which MATE does not need at all.

The most closely related works summarize the history with permutation-invariant aggregation. AMRL and SplAgger use it in end-to-end meta-RL but keep a recurrent branch (Beck et al., 2020; 2024). Task-inference methods use invariant context encoders more often, such as the product of Gaussian factors in PEARL, the Deep Sets sum in ELUE, the pooled transition embeddings in FOCAL and CORRO, and the closed-form Bayesian posterior in GLiBRL (Rakelly et al., 2019; Imagawa et al., 2022; Zaheer et al., 2017; Li et al., 2021; Yuan & Lu, 2022; You & Kurniawati, 2025), but each keeps the aggregate inside an explicit inference pipeline. In MATE, permutation-invariant aggregation is the *only* memory, with no recurrent backbone, attention, or inference objective, and our sufficiency analysis shows that it still retains enough information for the optimal CMDP policy.

2.3 Subset Training of Memory-Based Agents

Memory-based agents are almost always trained with a contiguous subset of the episode, and gradient stops at the subset boundary, which caps the memory horizon at the subset size (Williams & Peng, 1990; Ni et al., 2022; Grigsby et al., 2024b; Morad et al., 2024). Existing remedies each pay a price: stored recurrent states with burn-in are stale and still truncate the gradient (Kapturowski et al., 2019), full-episode training costs an update proportional to the episode length (Lu et al., 2023; Morad et al., 2024), and segment-level recurrence passes cached states or memory tokens across subsets but gives the cache no gradient (Dai et al., 2019; Cherepanov et al., 2026). Outside RL, UMBC trains additive set encoders through a random subset of each set with a rescaled loss that keeps the gradient unbiased, but recomputes the other elements at every update (Willette et al., 2023). STORE brings this estimator to memory-based RL and reuses the embeddings stored at rollout instead of recomputing them, so every step still sees the whole history for the price of a subset update and gradient reaches any earlier transition.

3 Preliminaries

3.1 Contextual MDP

A Contextual Markov Decision Process (CMDP) (Hallak et al., 2015) is defined as a set of MDPs $\{(\mathcal{S}, \mathcal{A}, P_c) \mid c \in \mathcal{C}\}$, where $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{C}$ denote the state, action, and context spaces. The *unobserved* context c determines the transition kernel P_c, given by $p(s_{t+1}, r_t | s_t, a_t, c)$. We call $x_t = (s_{t-1}, a_{t-1}, r_{t-1}, s_t)$ the *transition* at time t and write $x_{1:t}$ for the transition history. A CMDP can be viewed as a special case of a partially observable MDP (POMDP) (Cassandra et al., 1994) where the unobserved state is the context.

The goal in a CMDP is to find an optimal policy π^* that maximizes the expected discounted return under the context prior $p(c)$, $\max_\pi \mathbb{E}_{c\sim p(c),\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$, where γ is the discount factor. Thus, the optimal agent generalizes across MDPs induced by different contexts and adapts to the current MDP within an episode.

For CMDPs, the optimal policy and value function can be expressed as functions of the current state s_t and the posterior belief over contexts given the transition history, $p(c|x_{1:t})$:

$$\pi^*(a_t | s_t, p(c|x_{1:t})), \quad V^*(s_t, p(c|x_{1:t})). \tag{1}$$

This formulation is derived by casting the CMDP as an MDP with an augmented state $s_t^+ = (s_t, p(c|x_{1:t}))$ Duff (2002); Ross et al. (2007); Zintgraf et al. (2020). However, exact posterior inference is typically intractable, as it requires knowledge of the unknown transition kernel P_c and may not admit a finite-dimensional representation.

3.2 Memory-based RL

To learn history-dependent policies as required in CMDPs (Equation (1)), leveraging sequence models to compress the transition history into a latent memory has emerged as a promising approach (Ni

Algorithm 1 STORE update for MATE. ℓ^{RL} is the loss of the base off-policy algorithm, with each state input s_t replaced by (s_t, m_t) as in Equation (2).

Given: episode $x_{1:T}$, embeddings $\bar{z}_{1:T}$ stored at rollout

1: Sample independent size-k subsets S (transitions to re-embed) and S' (loss steps) of $\{1, \dots, T\}$
2: $z_i \leftarrow E_\psi(x_i)$ for $i \in S$, $z_i \leftarrow \bar{z}_i$ for $i \notin S$
3: $m_t \leftarrow \sum_{i \le t} z_i$ for $t = 0, \dots, T$
4: $\hat{\mathcal{L}}_{S'} \leftarrow \frac{1}{k} \sum_{t \in S'} \ell_t^{\mathrm{RL}}$ with $\ell_t^{\mathrm{RL}} = \ell^{\mathrm{RL}}\big((s_{t-1}, m_{t-1}), a_{t-1}, r_{t-1}, (s_t, \mathrm{sg}(m_t))\big)$
5: Update all parameters with $\nabla \hat{\mathcal{L}}_{S'}$, where the encoder gradient $\nabla_\psi \hat{\mathcal{L}}_{S'}$ is scaled by $\beta = T/k$
6: Refresh stored embeddings, $\bar{z}_i \leftarrow \mathrm{sg}(z_i)$ for $i \in S$

Table 1: **MATE combines linear cost with parallelizable updates.** Total complexity of rollout and update over an episode of length T for MATE, causal Transformers, and RNNs, and whether the update is parallelizable across the sequence dimension.

Model	Rollout	Update	Parallelizable
Transformers	$\mathcal{O}(T^2)$	$\mathcal{O}(T^2)$	**Yes**
RNNs	$\mathcal{O}(\boldsymbol{T})$	$\mathcal{O}(\boldsymbol{T})$	No
MATE	$\mathcal{O}(\boldsymbol{T})$	$\mathcal{O}(\boldsymbol{T})$	**Yes**

et al., 2022; Beck et al., 2023; Grigsby et al., 2024a;b). In this work, we refer to this general framework as **memory-based RL**.

Memory-based RL employs a memory network m_ψ, parametrized by ψ, that summarizes the transition history into a finite-dimensional memory $m_t = m_\psi(x_{1:t}) \in \mathbb{R}^m$. This is then used to condition the memory-augmented policy and value networks:

$$\pi_\theta(a_t | s_t, m_t), \quad V_\phi(s_t, m_t). \tag{2}$$

All networks are jointly trained with RL. Existing methods vary in the critic formulation (V_ϕ, Q_ϕ, or none) and in how the networks are conditioned (on (s_t, m_t) or on m_t alone). We use Equation (2) as a representative case in our analysis.

Transformers Grigsby et al. (2024a;b); Shala et al. (2025) and RNNs Duan et al. (2016); Ni et al. (2022); Beck et al. (2023), or their variants Beck et al. (2020; 2024); Huang et al. (2024), are the standard choices for m_ψ. RNNs offer the advantage of constant inference costs during rollout, but their recurrent nature often leads to gradient instabilities during the update step Bengio et al. (1994); Pascanu et al. (2013). Transformers excel at long-term memorization Ni et al. (2023), but their reliance on attention mechanisms incurs linearly growing inference costs per rollout step.

4 Memory of Accumulated Transition Embeddings

In this section, we introduce **M**emory of **A**ccumulated **T**ransition **E**mbeddings (**MATE**), a simple sum-aggregated memory architecture for CMDPs that addresses the limitations of Transformers and RNNs. MATE builds on the permutation invariance of the context posterior established in Section 4.1. We first describe the detailed architecture of MATE and its computational advantages over Transformers and RNNs in Section 4.2. In Section 4.3, we establish that MATE provides a theoretically grounded representation for solving CMDPs.

4.1 Permutation Invariance in CMDPs

The permutation invariance of the context posterior $p(c|x_{1:t})$ is a fundamental property in CMDPs that serves as both the primary motivation and the theoretical foundation for our architecture. We therefore explicitly formalize this property in Equation (4), noting its utilization in prior studies (Imagawa et al., 2022; Beck et al., 2024). By applying Bayes' rule and leveraging the conditional independence of transitions given c, the posterior factorizes as:

$$p(c|x_{1:t}) \propto p(c) \prod_{i=0}^{t-1} p(s_{i+1}, r_i | s_i, a_i, c), \tag{3}$$

assuming that s_0 is independent of c (see Appendix A.1 for full derivation). Since the product operation in Equation (3) is commutative, the posterior is invariant to the order of transitions. Thus, for any permutation $\rho : \{1, \dots, t\} \to \{1, \dots, t\}$,

$$p(c|x_1, \dots, x_t) = p(c|x_{\rho(1)}, \dots, x_{\rho(t)}). \tag{4}$$

4.2 Memory Architecture

A desirable memory m_t for CMDPs should act as a tractable surrogate for the context posterior $p(c|x_{1:t})$, as suggested by comparing Equation (1) with Equation (2). Motivated by the permutation invariance of $p(c|x_{1:t})$ in Equation (4), we define MATE as a permutation-invariant memory

$$m_t = \sum_{i=1}^{t} E_\psi(x_i). \tag{5}$$

The transition encoder E_ψ can be any network that maps each transition x_i to an embedding. As illustrated in Figure 6 in Appendix B.1, MATE summarizes the transition history through simple accumulation, unlike Transformer or RNN-based memories that utilize attention or recurrent layers. When the posterior is a product of Gaussian factors as in Rakelly et al. (2019), this sum of suitably chosen embeddings encodes the posterior exactly (Appendix A.2)

MATE is cheaper to run than causal Transformers and RNNs, the two standard memory architectures in memory-based RL (Table 1). Specifically, MATE maintains a constant inference cost per rollout step via the update rule $m_{t+1} = m_t + E_\psi(x_{t+1})$. In contrast, causal Transformers incur an $\mathcal{O}(t)$ cost per step due to the growing attention window. MATE also supports parallel update across the sequence dimension, a capability lacking in RNNs (Lu et al., 2023). We note that while bidirectional Transformers could theoretically offer permutation invariance, they must re-encode the full history at every rollout step. To the best of our knowledge, prior memory-based RL work has exclusively adopted causal Transformers.

4.3 Sufficiency of MATE for Optimal Policies

MATE makes the policy independent of the transition order, so it cannot represent arbitrary history-dependent policies. This constraint is harmless in CMDPs, because the optimal policy depends on the history only through the posterior, which is order invariant (Equation (4)). Assume that the transition space $\mathcal{X} \subset \mathbb{R}^n$ and the state space $\mathcal{S} \subset \mathbb{R}^{d_s}$ are compact, and let $\mathcal{M}_{\leq T}(\mathcal{X})$ be the set of multisets of at most T elements of $\mathcal{X}$. We can then write the optimal policy of Equation (1) as a map $\pi^* : \mathcal{S} \times \mathcal{M}_{\leq T}(\mathcal{X}) \to \mathcal{Y} \subset \mathbb{R}^{d_y}$, where $\mathcal{Y}$ holds the parameters of the action distribution, such as action probabilities or the mean and variance of a Gaussian. A one-layer transition encoder already suffices to represent this policy.

Proposition 4.1. *Let $E_\psi(x) = f(Ax + b)$ be a one-layer MLP with parameters $\psi = (A, b)$, output dimension $m = 2nT + 1$, and an analytic non-polynomial activation f. If π^* is continuous, then for Lebesgue-almost-every (A, b) there exists a continuous $\pi : \mathcal{S} \times \mathbb{R}^m \to \mathbb{R}^{d_y}$ such that $\pi\big(s_t, \sum_{i=1}^{t} E_\psi(x_i)\big) = \pi^*(s_t, p(c|x_{1:t}))$ for every history $x_{1:t}$ with $t \leq T$.*

Proof sketch. For almost every (A, b), the sum map $\hat{E}(\{\{x_1, \ldots, x_t\}\}) = \sum_{i=1}^{t} E_\psi(x_i)$ is injective on $\mathcal{M}_{\leq T}(\mathcal{X})$ (Amir et al., 2023, Theorem 3.3). As $\mathcal{M}_{\leq T}(\mathcal{X})$ is compact, $\hat{E}$ is a homeomorphism onto its closed image $\mathcal{Z}$, so $\pi(s_t, z) = \pi^*(s_t, \hat{E}^{-1}(z))$ is continuous on $\mathcal{S} \times \mathcal{Z}$ and extends continuously to $\mathcal{S} \times \mathbb{R}^m$ by the Tietze extension theorem (Appendix A.3).

This existence result directly extends to the optimal value function with the same choice of E_ψ, which justifies the architectural choice of sharing the memory m_t between the policy and value networks (Corollary A.2).

In practice, we use neural networks to approximate π and V and train them end-to-end together with the transition encoder E_ψ, as in standard memory-based RL (Equation (2)). Since π and V are continuous, the universal approximation theorem (Cybenko, 1989) implies that these networks can approximate the optimal policy and value function to arbitrary accuracy. The sum m_t itself, however, is a poor network input, because its norm grows with t. We therefore feed the networks $\hat{m}_t = (m_t + \Psi)/\|m_t + \Psi\|$, the memory shifted by a learned offset Ψ and projected onto the unit hypersphere. With $m = 2nT + 2$ and almost every Ψ, Proposition 4.1 and Corollary A.2 still hold with $\hat{m}_t$ in place of m_t (Appendix A.4). The memory dimension also need not reach $2nT + 2$, as MATE attains its full performance with a much smaller memory (Figure 7). This suggests that the learned memory retains only the information relevant to the context posterior, rather than the entire transition history.

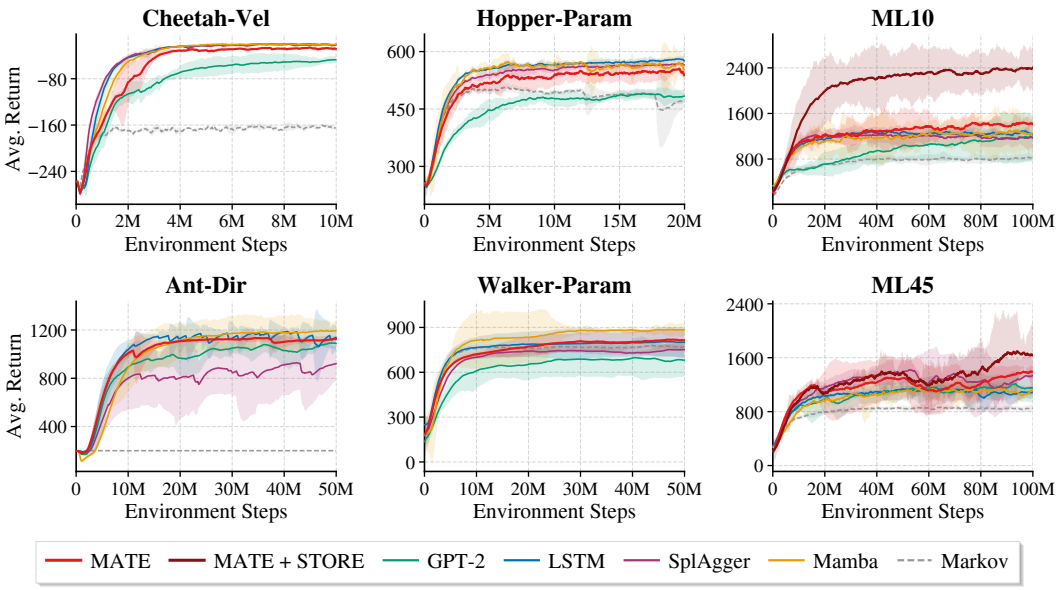


Figure 2: **Learning curves on MuJoCo (Left, Center) and Meta-World (Right).** Tasks are ordered by increasing difficulty from top to bottom. All agents are trained with SAC Haarnoja et al. (2018). Curves are smoothed with an exponential moving average. Solid lines and shaded regions show the mean and the 95% Student's t confidence interval over 5 seeds.

Table 2: **Training time for the Figure 2 experiments.** Measured in hours on NVIDIA V100 GPU, median over seeds. Memory Baselines share the memory dimension and depth. Markov (gray) is a memory-free reference with a $2\times$ wider state embedder.

	Cheetah-Vel	Hopper-Param	Ant-Dir	Walker-Param	ML10	ML45
MATE	**4.2**	**7.8**	**22.1**	**19.3**	**91.7**	**127.7**
(+STORE)	+ 0.0	+0.0	+0.5	+0.2	+0.9	+1.5
GPT-2	4.5	9.5	31.5	33.2	114.2	198.4
LSTM	4.7	12.0	35.1	34.8	149.8	231.7
SplAgger	5.0	11.0	37.1	35.7	146.7	223.9
Mamba	6.3	14.2	38.3	40.5	146.3	193.6
Markov	3.8	7.3	27.0	24.1	96.6	145.2

5 Subset Training over Reused Embeddings

Memory-based agents are usually trained on a contiguous subset of $k < T$ transitions per episode (Williams & Peng, 1990; Ni et al., 2022; Grigsby et al., 2024b). This subset training is what keeps training on long episodes affordable, since the cost and memory footprint of an update then scale with k rather than T for every architecture in Table 1. However, no gradient crosses the boundary of the subset, so the memory horizon cannot exceed k (Morad et al., 2024). This limit is fatal on T-Maze, where the agent must remember a cue observed at the start of the episode until the decision at its end (Figure 8).

Subset **T**raining **O**ver **R**eused **E**mbeddings (**STORE**) is a training algorithm for MATE that removes this limit on the memory horizon while keeping the update cost of subset training (Algorithm 1). Each update recomputes the embeddings of a random subset S of transitions, reuses stored embeddings for the rest, and evaluates the RL loss at an independent random subset S' of steps. We take $|S| = |S'| = k$, which matches the cost of subset training, where the same k contiguous steps serve both roles. Because the subsets are random rather than contiguous, a loss step is trained together with re-embedded transitions at any distance, not only within k steps. The agent thus learns to act on transitions observed arbitrarily far in the past, which lifts the cap on the memory horizon. Random subsets are possible because the memory of MATE is a sum of per-transition embeddings, so the embedding of any single transition can be replaced without touching the others. A stored recurrent

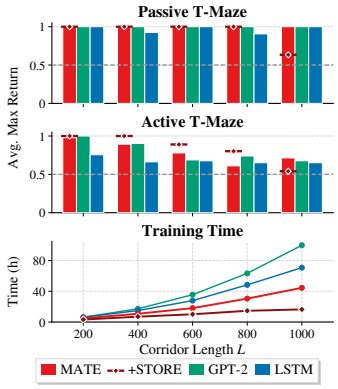


Figure 3: **Performance and training cost on T-Maze.** The top two rows show the maximum evaluation return reached during training on Passive and Active T-Maze with corridor length L, averaged over 5 seeds as in Grigsby et al. (2024a). The dashed line is the upper bound of a Markov agent. All agents are trained with DDQN (Van Hasselt et al., 2016). The bottom row shows the training time on Passive T-Maze.

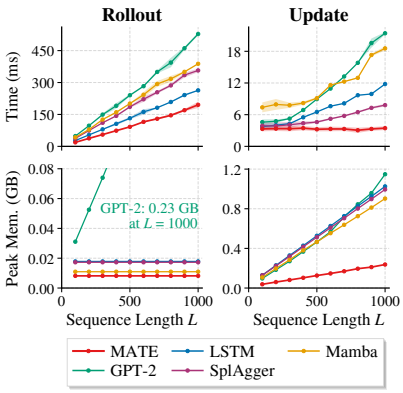


Figure 4: **Compute cost of the memory module.** Wall-clock time and peak memory of the memory module alone, measured on one V100 GPU as a function of sequence length L. Lines show the mean and shading the standard deviation over 30 runs. MATE is the cheapest at every L in all four panels. See Appendix B.3 for the setup.

state, by contrast, depends on every earlier transition through the recursion and can only be reused as a contiguous prefix (Kapturowski et al., 2019). Proposition 5.1 shows that the resulting update is an unbiased estimate of the full-episode gradient when the stored embeddings are exact.

Proposition 5.1. *Let $\mathcal{L}$ be the full-episode loss of Algorithm 1, obtained with $S = S' = \{1, \dots, T\}$, and let ϕ denote all trainable parameters other than ψ. Let S and S' be independent and uniform over the size-k subsets of $\{1, \dots, T\}$, $1 \le k \le T$, and assume $\bar{z}_i = \mathrm{sg}(E_\psi(x_i))$ for all i, where* sg *is the stop-gradient operator. Then $\mathbb{E}[\nabla_\phi \hat{\mathcal{L}}_{S'}] = \nabla_\phi \mathcal{L}$ and $\mathbb{E}[\beta \nabla_\psi \hat{\mathcal{L}}_{S'}] = \nabla_\psi \mathcal{L}$ with $\beta = T/k$.*

Intuitively, when the stored embeddings are exact, the choice of S and S' changes no value in the forward pass and only decides which embeddings carry gradient and at which steps the loss is evaluated. Given a loss step, each earlier transition is re-embedded with probability k/T, so scaling the encoder gradient by $\beta = T/k$ restores the expectation of the full-episode gradient (see Appendix A.5 for the proof). In practice the assumption holds only approximately, since the stored embeddings come from earlier encoders. To limit this staleness, each episode draws S as consecutive blocks of a random permutation of its indices, so every stored embedding is refreshed once per pass of $\lceil T/k \rceil$ updates on that episode.

6 Experiments

We evaluate MATE on three CMDP benchmarks, MuJoCo, Meta-World, and T-Maze. Section 6.3 adds two further settings that tests its extensibility. We detail the experimental setup in Section 6.1, followed by main results in Section 6.2. Additional details are provided in Appendix B.

6.1 Experimental Setup

Benchmarks. The benchmarks differ in what the hidden context controls (Appendix C). *MuJoCo* (Todorov et al., 2012) has four meta-RL locomotion tasks with $T = 200$ (Finn et al., 2017; Rakelly et al., 2019; Li et al., 2024). Ant-Dir and Cheetah-Vel hide the target direction or speed and vary the reward, while Walker-Param and Hopper-Param hide the masses, damping, and friction of the body and vary the dynamics. *Meta-World* ML10 and ML45 (Yu et al., 2020) are manipulation suites with $T = 500$ in which each episode samples one of 10 or 45 tasks with its goal and object positions, none of which the agent observes. In *T-Maze* (Ni et al., 2023), reformulated as a CMDP (Appendix C.3), the agent must recall a goal direction at the end of a corridor of length L from 200 to 1000. The

Passive variant shows the direction at the start and tests the memory horizon alone. The *Active* variant requires an exploratory step to reveal it and also tests credit assignment.

Baselines. We compare MATE, whose transition encoder is a plain MLP, with GPT-2 (Radford et al., 2019) and LSTM (Hochreiter & Schmidhuber, 1997) as the standard Transformer and RNN memories, following Ni et al. (2023). We also include two architectures that each share a design principle with MATE. SplAgger (Beck et al., 2024) adds a permutation-invariant max aggregation to a GRU, and the state-space model Mamba (Gu & Dao, 2024) combines constant per-step rollout cost with a parallel update. A memory-free Markovian policy serves as a reference for the value of memory, and we report its optimal return where known. All agents share the architecture of Figure 5. A one-layer MLP projects each transition to $\mathbb{R}^m$, the memory architecture maps the projected history to $m_t \in \mathbb{R}^m$, and an MLP embeds s_t into $\mathbb{R}^m$. Their concatenation feeds the actor and critic, trained with SAC (Haarnoja et al., 2018) for continuous actions and DDQN (Van Hasselt et al., 2016) for discrete ones. The Markovian policy uses a $2m$-wide state embedding in place of the memory. Memory agents train on a contiguous subset of k transitions per episode (Section 5), with $k = 32$ on MuJoCo, $k = 128$ on Meta-World, and $k = T$ on T-Maze. MATE with STORE uses the same k except $k = 100$ on T-Maze. Table 3 lists the memory dimension, depth, and budget per environment.

6.2 Main Results

MATE matches the baselines at a lower training cost. On MuJoCo and Meta-World, MATE matches the return of the memory baselines, and no single model is best on every task (Figure 2). On MuJoCo, the returns of most agents saturate at a similar level, and MATE reaches this level as well. Mamba attains the highest return on the two harder tasks, Ant-Dir and Walker-Param, while LSTM and SplAgger, both built on a recurrent network, train unstably on Ant-Dir. On ML10 and ML45, MATE attains a slightly higher return than every memory baseline. MATE also trains the fastest of the memory models on all six tasks, and its training time stays close to that of the memory-free Markov policy (Table 2).

On T-Maze, where we compare with GPT-2 and LSTM following Ni et al. (2023), MATE matches the stronger of the two, GPT-2 (Figure 3). Both reach the maximum return of 1 on Passive T-Maze at every corridor length L up to 1000 and stay close on Active T-Maze, each ahead at some lengths. LSTM falls slightly short on Passive T-Maze and fails to reach the maximum return on Active T-Maze already at $L = 200$. MATE also trains faster than both at every L (bottom row of Figure 3). The training time of GPT-2 grows the most with L, reflecting the quadratic cost of attention (Table 1), so GPT-2 becomes the slowest at large L. Timing the memory module alone confirms that the saving comes from the memory (Figure 4). MATE uses the least time and peak memory at every sequence length, in both rollout and update, and the gap widens as the sequence grows.

STORE extends the memory horizon and improves the return. On T-Maze, we train MATE + STORE on subsets of $k = 100$ transitions, and a contiguous subset of this size would cap the memory horizon at 100 steps (Section 5). Yet MATE + STORE reaches the maximum return on Passive T-Maze up to $L = 800$, eight times the subset size, and falls below it only at $L = 1000$ (Figure 3). The small subsets also cut the cost of training. At $L = 1000$, each update re-embeds about a tenth of the transitions and evaluates the loss at about a tenth of the steps, so MATE + STORE trains $6.1\times$ faster than GPT-2 (bottom row of Figure 3).

Besides extending the memory horizon, STORE also improves the return of MATE. On ML10 and ML45, where the returns have not saturated, MATE + STORE clearly outperforms MATE trained on contiguous subsets of the same size $k = 128$ (Figure 2), and it adds only about 1% to the training time (Table 2). On Active T-Maze, it even attains a higher return than MATE trained on full episodes at $L = 400$, 600, and 800. A likely reason for the gain on Meta-World is that STORE removes a mismatch between training and rollout. With contiguous subsets, the memory in an update is built only from the transitions inside the subset, whereas at rollout the agent acts on a memory built from its whole history of up to $T = 500$ transitions. STORE builds the memory in an update from the whole history, as at rollout.

6.3 Ablations and Extensions

Effect of the offset Ψ. The offset Ψ lets the normalized memory distinguish histories that differ only in scale (Appendix A.4). Without it, $\hat{m}_t = m_t/\|m_t\|$ maps $\{\{x\}\}$ and $\{\{x, x\}\}$ to the same point, so the agent cannot tell how many times a transition has been observed (Figure 9).

Compatibility with auxiliary objective. Recent work supplements the RL loss with an auxiliary objective when the reward alone does not shape a useful representation. To show that MATE composes with this practice, we run Alchemy with an InfoNCE (Oord et al., 2018) based auxiliary loss on the memory, described in Appendix B.4. No memory architecture we train, MATE included, separates meaningfully from a Markov policy, indicating that the RL loss on its own is insufficient to shape memory in any of them. With the auxiliary objective, MATE clears that reference by a wide margin, showing that MATE is also compatible with training under auxiliary objectives (Figure 10).

Extension to Visual Observation Task. MATE places no restriction on the transition encoder, so it extends to pixel observations by prepending a CNN, shared by all agents. We test this on Vehicle Racing from CARL (Benjamins et al., 2023): the agent drives one of 29 vehicles on one of 10 tracks from 96×96 pixel observations, and the context (vehicle type and track) is only partially visible, since neither the drivetrain nor the course beyond the camera is rendered (Appendix C.4). Figure 11 shows that MATE matches the baselines.

7 Conclusion

We presented MATE, a memory architecture for CMDPs that represents the transition history as a sum of transition embeddings. This design follows from the order invariance of the context posterior, and we proved that it can still represent the optimal policy and value function. Each rollout step updates the memory in constant time, and each update step computes it in parallel across the sequence. The additive memory also enables STORE, which re-embeds a random subset of transitions and reuses stored embeddings for the rest. STORE keeps the cost of subset training and places no limit on the memory horizon, and its update is an unbiased estimate of the full-episode gradient when the stored embeddings are up to date. On MuJoCo, Meta-World, and T-Maze, MATE performs on par with GPT-2, LSTM, SplAgger, and Mamba while training in the least time on every task. With STORE, MATE learns dependencies far longer than its subset size on T-Maze and reaches a higher return on Meta-World. MATE also works with an auxiliary contrastive objective and with pixel observations.

MATE assumes that the context stays fixed within an episode and that transitions are conditionally independent given it. When the context drifts within an episode, or when the order of observations carries information as in general POMDPs, a plain sum discards information the agent needs. Weighted or decaying sums could relax this assumption while remaining linear in the embeddings and thus compatible with STORE. The guarantee of STORE also holds only approximately in practice, since the stored embeddings come from earlier encoders. We limit this staleness with a refresh schedule but do not bound the bias it leaves. Our T-Maze comparison omits SplAgger and Mamba because of our compute budget, and all of our evidence comes from simulation. Finally, on Alchemy the RL loss alone fails to shape any memory we tested, while an auxiliary objective built from the partial sums of MATE succeeds at little extra cost. Combining MATE with deep self-supervised RL (Wang et al., 2025) is therefore a promising route toward harder tasks and toward the real-world settings, such as sim-to-real transfer, that motivate this work.

AI use statement

The method, the theoretical results, and the choice of what to investigate are our own, and we used generative AI tools as an assistant within that work. Among the categories requiring disclosure, we used AI tools to suggest and refine hypotheses, to propose experiment designs and give feedback on methodology, and to produce initial interpretations of results. In each case we decided what to pursue and treated the output as a suggestion rather than a conclusion. We did not use generative AI tools to generate synthetic data sets, to develop the theoretical models of this paper, to formulate its mathematical claims, to provide ingredients for their proofs, or to assist in writing those proofs. Cleaning or reformatting of datasets, translation, and support for qualitative and thematic analysis are not applicable to this work.

Among the categories where disclosure is recommended, we used AI tools to suggest experimental parameters, to write and edit software for launching, monitoring, and aggregating experiments, to draft parts of the appendix and of this section, to improve the readability of parts of the text, for brainstorming, and to identify relevant literature and format references. We have reviewed all AI-assisted work. We adopted hypotheses proposed with AI assistance only after examining the evidence, and we withdrew several when further runs contradicted them. We executed all AI-written code and checked its outputs against the logged training records. Every number in this paper is traceable to a training run whose configuration and complete evaluation history we recorded, and no reported result was produced by a generative model. We edited all AI-assisted text, and we take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

Ethics statement

This work studies a memory architecture for contextual Markov decision processes. All experiments are conducted in simulation, on T-Maze, MuJoCo locomotion tasks, Meta-World, and Symbolic Alchemy. No human subjects were involved, no personal or sensitive data were collected or used, and no new data were released. The environments are publicly available research benchmarks and were used in accordance with their licenses.

The contribution is an architectural one and carries no direct application, but it is intended to make agents that adapt online to an unobserved context more practical, and the settings we cite as motivation include physical domains such as robotics, autonomous driving, and healthcare. Methods that improve online adaptation in those domains can fail in ways that matter, and the evidence we provide is confined to simulated benchmarks with known context distributions. We make no claim about behavior under distribution shift, afoubt safety, or about reliability outside evaluated settings, and we would regard deployment in a safety critical system as requiring separate analysis that this paper does not supply.

Reproducibility statement

The memory architecture is defined in Section 4.2 and the normalization applied to it in Appendix A.4, and the STORE training algorithm in Section 5 and Algorithm 1. The assumptions behind Proposition 4.1 and Corollary A.2, and Proposition 5.1 are stated where each result is introduced, an the corresponding derivations and full proofs are given in Appendix A.1 through Appendix A.5. Network architectures, optimizer settings, and all hyperparameters for MATE and for every baseline are listed in Appendix B, and benchmarks, their context distributions, and their per task budgets in Appendix C. Section 6.1 describes the protocol shared by all experiments, and every reported curve is the mean over three seeds under a budget matched across architectures. Source code will be made publicly available upon publication.

References

Tal Amir, Steven Gortler, Ilai Avni, Ravina Ravina, and Nadav Dym. Neural injective functions for multisets, measures and graphs via a finite witness theorem. *Advances in Neural Information Processing Systems*, 36:42516–42551, 2023.

Jacob Beck, Kamil Ciosek, Sam Devlin, Sebastian Tschiatschek, Cheng Zhang, and Katja Hofmann. Amrl: Aggregated memory for reinforcement learning. In *International Conference on Learning Representations*, 2020.

Jacob Beck, Risto Vuorio, Zheng Xiong, and Shimon Whiteson. Recurrent hypernetworks are surprisingly strong in meta-rl. *Advances in Neural Information Processing Systems*, 36:62121–62138, 2023.

Jacob Beck, Matthew Thomas Jackson, Risto Vuorio, Zheng Xiong, and Shimon Whiteson. Splagger: Split aggregation for meta-reinforcement learning. *CoRR*, 2024.

Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A tutorial on meta-reinforcement learning. *Foundations and Trends in Machine Learning*, 18(2-3):224–384, 2025.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Carolin Benjamins, Theresa Eimer, Frederik Schubert, Aditya Mohan, Sebastian Döhler, André Biedenkapp, Bodo Rosenhahn, Frank Hutter, and Marius Lindauer. Contextualize me–the case for context in reinforcement learning. *Transactions on Machine Learning Research*, 2023.

Anthony R Cassandra, Leslie Pack Kaelbling, and Michael L Littman. Acting optimally in partially observable stochastic domains. In *Aaai*, volume 94, pp. 1023–1028, 1994.

Egor Cherepanov, Alexey Staroverov, Alexey K Kovalev, and Aleksandr I Panov. Recurrent action transformer with memory. In *International conference on learning representations*, 2026.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 2978–2988, 2019.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Michael O'Gordon Duff. *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes*. University of Massachusetts Amherst, 2002.

L.C. Evans and R.F. Gariepy. *Measure Theory and Fine Properties of Functions*. Studies in Advanced Mathematics. Taylor & Francis, 1991. ISBN 9780849371578. URL https://books.google.co.kr/books?id=HZONacVPGlMC.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135. PMLR, 06–11 Aug 2017.

Jake Grigsby, Jim Fan, and Yuke Zhu. Amago: Scalable in-context reinforcement learning for adaptive agents. In *International Conference on Learning Representations*, volume 2024, pp. 26919–26952, 2024a.

Jake Grigsby, Justin Sasek, Samyak Parajuli, Daniel Adebi, Amy Zhang, and Yuke Zhu. Amago-2: Breaking the multi-task barrier in meta-reinforcement learning with transformers. *Advances in Neural Information Processing Systems*, 37:87473–87508, 2024b.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First conference on language modeling*, 2024.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015.

Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado Van Hasselt. Multi-task deep reinforcement learning with popart. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3796–3803, 2019.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Sili Huang, Jifeng Hu, Zhejian Yang, Liwei Yang, Tao Luo, Hechang Chen, Lichao Sun, and Bo Yang. Decision mamba: Reinforcement learning via hybrid selective sequence modeling. *Advances in Neural Information Processing Systems*, 37:72688–72709, 2024.

Takahisa Imagawa, Takuya Hiraoka, and Yoshimasa Tsuruoka. Off-policy meta-reinforcement learning with belief-based task inference. *IEEE Access*, 10:49494–49507, 2022.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*, 2019.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76:201–264, 2023.

Lanqing Li, Rui Yang, and Dijun Luo. FOCAL: Efficient fully-offline meta-reinforcement learning via distance metric learning and behavior regularization. In *International conference on learning representations*, 2021.

Lanqing Li, Hai Zhang, Xinyu Zhang, Shatong Zhu, Yang Yu, Junqiao Zhao, and Pheng-Ann Heng. Towards an information theoretic framework of context-based offline meta-reinforcement learning. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, pp. 75642–75667, 2024.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

Chris Lu, Yannick Schroecker, Albert Gu, Emilio Parisotto, Jakob Foerster, Satinder Singh, and Feryal Behbahani. Structured state space models for in-context reinforcement learning. *Advances in Neural Information Processing Systems*, 36:47016–47031, 2023.

Steven Morad, Ryan Kortvelesy, Stephan Liwicki, and Amanda Prorok. Reinforcement learning with fast and forgetful memory. *Advances in neural information processing systems*, 36, 2023.

Steven Morad, Chris Lu, Ryan Kortvelesy, Stephan Liwicki, Jakob Foerster, and Amanda Prorok. Recurrent reinforcement learning with memoroids. *Advances in Neural Information Processing Systems*, 37:14386–14416, 2024.

Tianwei Ni, Benjamin Eysenbach, and Ruslan Salakhutdinov. Recurrent model-free rl can be a strong baseline for many pomdps. In *International Conference on Machine Learning*, pp. 16691–16723. PMLR, 2022.

Tianwei Ni, Michel Ma, Benjamin Eysenbach, and Pierre-Luc Bacon. When do transformers shine in rl? decoupling memory from credit assignment. *Advances in Neural Information Processing Systems*, 36:50429–50452, 2023.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pp. 1310–1318. Pmlr, 2013.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pp. 5331–5340. PMLR, 2019.

Stephane Ross, Brahim Chaib-draa, and Joelle Pineau. Bayes-adaptive pomdps. *Advances in neural information processing systems*, 20, 2007.

Gresa Shala, André Biedenkapp, Pierre Krack, Florian Walter, and Josif Grabocka. Efficient cross-episode meta-rl. In *The Thirteenth International Conference on Learning Representations*, 2025.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

Mark Towers, Ariel Kwiatkowski, John Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Kallinteris Andreas, Markus Krimmel, Arjun Kg, Rodrigo Perez-Vicente, et al. Gymnasium: A standard interface for reinforcement learning environments. *Advances in Neural Information Processing Systems*, 38, 2026.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Jane X Wang, Michael King, Nicolas Porcel, Zeb Kurth-Nelson, Tina Zhu, Charlie Deck, Peter Choy, Mary Cassin, Malcolm Reynolds, Francis Song, et al. Alchemy: A benchmark and analysis toolkit for meta-reinforcement learning agents. *arXiv preprint arXiv:2102.02926*, 2021.

Kevin Wang, Ishaan Javali, Michal Bortkiewicz, Tomasz Trzcinski, and Benjamin Eysenbach. 1000 layer networks for self-supervised rl: Scaling depth can enable new goal-reaching capabilities. *CoRR*, 2025.

Jeffrey Willette, Seanie Lee, Bruno Andreis, Kenji Kawaguchi, Juho Lee, and Sung Ju Hwang. Scalable set encoding with universal mini-batch consistency and unbiased full set gradient approximation. In *International conference on machine learning*, pp. 37008–37041. PMLR, 2023.

Ronald J Williams and Jing Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural computation*, 2(4):490–501, 1990.

Jingyang You and Hanna Kurniawati. Meta-RL with Bayesian linear task models. *arXiv preprint arXiv:2512.20974*, 2025.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Haoqi Yuan and Zongqing Lu. Robust task representations for offline meta-reinforcement learning via contrastive learning. In *International conference on machine learning*, pp. 25747–25759. PMLR, 2022.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

L Zintgraf, K Shiarlis, M Igl, S Schulze, Y Gal, K Hofmann, and S Whiteson. Varibad: a very good method for bayes-adaptive deep rl via meta-learning. *Proceedings of ICLR 2020*, 2020.

A Proofs and Derivations

A.1 Factorization of the Context Posterior

Based on the CMDP assumptions that (i) the transition kernel satisfies the Markov property conditioned on c, and (ii) the policy depends only on the interaction history and not directly on the unobserved context, the trajectory distribution given c factorizes as:

$$p(x_{1:t}|c) = p(s_0)\pi(a_0|s_0)p(s_1, r_0|s_0, a_0, c)\dots\pi(a_{t-1}|x_{1:t-1})p(s_t, r_{t-1}|s_{t-1}, a_{t-1}, c) \quad (6)$$

$$= p(s_0)\prod_{i=0}^{t-1}\pi(a_i|x_{1:i})p(s_{i+1}, r_i|s_i, a_i, c), \quad (7)$$

where we assume that s_0 is independent of c. Applying Bayes' rule and discarding terms that do not depend on c (namely $p(s_0)$ and $\pi(a_i|x_{1:i})$), we obtain:

$$p(c|x_{1:t}) \propto p(c)p(x_{1:t}|c) \propto p(c)\prod_{i=0}^{t-1} p(s_{i+1}, r_i|s_i, a_i, c). \tag{8}$$

A.2 Probabilistic Interpretation of Sum Aggregation

To intuitively understand why sum aggregation can be a natural choice, consider a scenario where the context posterior $p(c|x_{1:t})$ is modeled as a product of Gaussian factors, following Rakelly et al. (2019). Under the assumption of a uniform prior $p(c)$, the posterior distribution takes the form:

$$p(c|x_{1:t}) \propto \prod_{i=1}^{t} \mathcal{N}(c; \mu(x_i), \sigma^2(x_i)) \propto \prod_{i=1}^{t} \exp\left(-\frac{(c-\mu(x_i))^2}{2\sigma^2(x_i)}\right) \propto \exp\left(-\frac{(c-\mu_{post})^2}{2\sigma^2_{post}}\right). \tag{9}$$

Here, the posterior precision $1/\sigma^2_{post}$ and the precision-weighted mean $\mu_{post}/\sigma^2_{post}$ are determined by the sum of individual precisions and precision-weighted means:

$$\frac{1}{\sigma^2_{post}} = \sum_{i=1}^{t} \frac{1}{\sigma^2(x_i)}, \quad \frac{\mu_{post}}{\sigma^2_{post}} = \sum_{i=1}^{t} \frac{\mu(x_i)}{\sigma^2(x_i)}. \tag{10}$$

If we define the transition embedder $E_\psi(x_i)$ as:

$$E_\psi(x_i) = \left(\frac{\mu(x_i)}{\sigma^2(x_i)}, \frac{1}{\sigma^2(x_i)}\right), \tag{11}$$

then the accumulated memory m_t becomes

$$m_t = \sum_{i=1}^{t} E_\psi(x_i) = \left(\frac{\mu_{post}}{\sigma^2_{post}}, \frac{1}{\sigma^2_{post}}\right). \tag{12}$$

Therefore, m_t exactly encodes the sufficient statistics of the posterior $p(c|x_{1:t})$.

A.3 Proof of Proposition 4.1

Before presenting the main proof, we first establish a technical lemma regarding the topological properties of the multiset space. This lemma guarantees the compactness of the domain and the continuity of the summation function, which are prerequisites for establishing the homeomorphism and applying the extension theorem.

Lemma A.1. *Let $\mathcal{X}$ be a compact subset of $\mathbb{R}^n$, and $E : \mathbb{R}^n \to \mathbb{R}^m$ be a continuous function. Then, the space of bounded multisets $\mathcal{M}_{\leq T}(\mathcal{X})$, endowed with the disjoint union topology, is compact. Furthermore, the summation function $\hat{E} : \mathcal{M}_{\leq T}(\mathcal{X}) \to \mathbb{R}^m$ defined by $\hat{E}(\{\{x_1, \ldots, x_t\}\}) = \sum_{i=1}^{t} E(x_i)$ is continuous.*

Proof. First, we establish the compactness of $\mathcal{M}_{\leq T}(\mathcal{X})$. Since $\mathcal{X}$ is compact, the product space $\mathcal{X}^t$ is compact for any $t \leq T$. The component $\mathcal{M}_t(\mathcal{X})$ is the image of $\mathcal{X}^t$ under the continuous canonical quotient map, hence it is compact. Consequently, the finite disjoint union $\mathcal{M}_{\leq T}(\mathcal{X}) = \bigsqcup_{t=0}^{T} \mathcal{M}_t(\mathcal{X})$ is compact.

Next, regarding the continuity of $\hat{E}$, observe that the map $(x_1, \ldots, x_t) \mapsto \sum_{i=1}^{t} E(x_i)$ defined on $\mathcal{X}^t$ is continuous and permutation-invariant. By the definition of the quotient topology, this map induces a continuous restriction of $\hat{E}$ on each component $\mathcal{M}_t(\mathcal{X})$. Since $\hat{E}$ is continuous on each component of the disjoint union, it is continuous on the entire domain. □

Proposition 4.1 *Let $E_\psi : \mathcal{X} \to \mathbb{R}^m$ be a one-layer MLP with output dimension $m = 2nT + 1$ and an analytic non-polynomial element-wise activation $f : \mathbb{R} \to \mathbb{R}$, defined as*

$$E_\psi(x) = f(Ax + b), \text{ where } \psi = (A, b). \tag{13}$$

Then, given a continuous optimal policy $\pi^* : \mathcal{S} \times \mathcal{M}_{\leq T}(\mathcal{X}) \to \mathcal{Y} \subset \mathbb{R}^{d_y}$, there exists a continuous function $\pi : \mathcal{S} \times \mathbb{R}^m \to \mathbb{R}^{d_y}$ such that

$$\pi\big(s_t, \sum_{i=1}^{t} E_\psi(x_i)\big) = \pi^*(s_t, p(c|x_{1:t})) \tag{14}$$

for Lebesgue-almost-every $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$.

Proof. Let $\hat{E} : \mathcal{M}_{\leq T}(\mathcal{X}) \to \mathbb{R}^m$ be the summation function defined as $\hat{E}(\{\{x_1, ..., x_t\}\}) = \sum_{i=1}^{t} E_\psi(x_i)$. According to (Amir et al., 2023, Theorem 3.3), assuming $m = 2nT + 1$, $\hat{E}$ is injective for Lebesgue-almost-every parameters A and b. From Lemma A.1, the domain $\mathcal{M}_{\leq T}(\mathcal{X})$ is compact and $\hat{E}$ is continuous. Since a continuous bijection from a compact space to a Hausdorff space is a homeomorphism onto its image, $\hat{E}$ is a homeomorphism between $\mathcal{M}_{\leq T}(\mathcal{X})$ and its image $\mathcal{Z} \coloneqq \hat{E}(\mathcal{M}_{\leq T}(\mathcal{X})) \subset \mathbb{R}^m$. This implies that the inverse map $\hat{E}^{-1} : \mathcal{Z} \to \mathcal{M}_{\leq T}(\mathcal{X})$ is well-defined and continuous.

Now define $g : \mathcal{S} \times \mathcal{Z} \to \mathcal{Y}$ by

$$g(s_t, z) = \pi^*(s_t, p(c|x_{1:t})), \quad \text{where } \{\{x_1, \ldots, x_t\}\} = \hat{E}^{-1}(z). \tag{15}$$

This definition is well-posed because the posterior $p(c|x_{1:t})$ depends only on the multiset of transitions by Equation (4). The continuity of g follows directly from the continuity of π^* and $\hat{E}^{-1}$. Since $\mathcal{M}_{\leq T}(\mathcal{X})$ is compact, its image $\mathcal{Z}$ is compact and therefore closed in $\mathbb{R}^m$. Because $\mathcal{S}$ is a compact subset of Euclidean space, $\mathcal{S} \times \mathbb{R}^m$ is a normal topological space, and $\mathcal{S} \times \mathcal{Z}$ is a closed subset of it. Hence, by the Tietze extension theorem applied component-wise to the codomain $\mathbb{R}^{d_y}$, there exists a continuous extension $\pi : \mathcal{S} \times \mathbb{R}^m \to \mathbb{R}^{d_y}$ such that $\pi(s_t, z) = g(s_t, z)$ for all $z \in \mathcal{Z}$. Therefore,

$$\pi\big(s_t, \sum_{i=1}^{t} E_\psi(x_i)\big) = g\big(s_t, \sum_{i=1}^{t} E_\psi(x_i)\big) = \pi^*(s_t, p(c|x_{1:t})), \tag{16}$$

which verifies the claim. □

Corollary A.2. *Under the same assumptions as Proposition 4.1, for a continuous optimal value function $V^* : \mathcal{S} \times \mathcal{M}_{\leq T}(\mathcal{X}) \to \mathbb{R}$, there exists a continuous function $V : \mathcal{S} \times \mathbb{R}^m \to \mathbb{R}$ such that*

$$V(s_t, \sum_{i=1}^{t} E_\psi(x_i)) = V^*(s_t, p(c|x_{1:t})) \tag{17}$$

A.4 Hyperspherical Memory Normalization

In memory-based RL, the memory m_t serves as an input to the policy and value networks. Here, the sum aggregation of MATE can be problematic, as the norm of m_t typically grows with t. This unbounded growth poses challenges for neural network training, as it induces internal covariate shift and destabilizes optimization (Ioffe & Szegedy, 2015). To address this issue, we normalize the memory before it enters the networks by projecting it onto a hypersphere, $\hat{m}_t = (m_t + \Psi)/\|m_t + \Psi\|$. We demonstrate that the guarantee in Proposition 4.1 remains valid with a memory dimension of $m = 2nT + 2$ (see Proposition A.3) for almost-every $\Psi \in \mathbb{R}^m$. The offset Ψ is a necessary component that prevents information loss during normalization; without it, distinct histories such as $\{\{x, x\}\}$ and $\{\{x\}\}$ would yield the same normalized memory. In our experiments, we treat Ψ as a trainable parameter. We also scale $\hat{m}_t$ by the square root of its dimension, adopting the strategy in RMSNorm (Zhang & Sennrich, 2019).

Proposition A.3. *Let E_ψ be as in Proposition 4.1, but with output dimension $m = 2nT + 2$. Then for Lebesgue-almost-every $\Psi \in \mathbb{R}^m$, the conclusion of Proposition 4.1 holds with the normalized memory $\hat{m}_t = (m_t + \Psi)/\|m_t + \Psi\|$ defined above in place of $\sum_{i=1}^{t} E_\psi(x_i)$.*

Proof. Proposition 4.1 holds with $m = 2nT+2$ as well, since Amir et al. (2023, Theorem 3.3) gives injectivity of the summation map $\hat{E}$ for any $m \geq 2nT + 1$ and the rest of its proof does not depend

on m. Fix (A, b) for which $\hat{E}$ is injective, and let π be the continuous function from Proposition 4.1, so that $\pi(s_t, m_t) = \pi^*(s_t, p(c|x_{1:t}))$.

We first exclude a null set of offsets. For a history X of size t, write $S_t(x_{1:t}) = \sum_{i=1}^{t} E_\psi(x_i)$, which is a smooth map on $\mathbb{R}^{nt}$ since E_ψ is analytic on $\mathbb{R}^n$. The normalized memory of X is undefined when $\Psi = -S_t(x_{1:t})$. Two histories X and Y of sizes t and s yield the same normalized memory when $\hat{E}(X) + \Psi = \lambda(\hat{E}(Y) + \Psi)$ for some $\lambda > 0$. If $\lambda = 1$, injectivity of $\hat{E}$ gives $X = Y$. Otherwise,

$$\Psi = \frac{S_t(x_{1:t}) - \lambda\, S_s(y_{1:s})}{\lambda - 1}, \qquad \lambda \in (0,1) \cup (1, \infty). \tag{18}$$

In both cases Ψ lies in the image of a smooth map defined on an open subset of $\mathbb{R}^d$ with $d \leq n(t+s)+1 \leq 2nT+1 < m$. Such an image is Lebesgue-null in $\mathbb{R}^m$, since the domain is a countable union of compact sets on which the map is Lipschitz continuous, and Lipschitz maps do not increase Hausdorff dimension (Evans & Gariepy, 1991, Section 2.4.1). Let $N_{A,b}$ be the union of these images over all $t, s \leq T$, which is a finite union of null sets and hence null. For every $\Psi \notin N_{A,b}$, the map $\hat{E}_\Psi(X) = (\hat{E}(X) + \Psi)/\|\hat{E}(X) + \Psi\|$ is well defined and injective on $\mathcal{M}_{\leq T}(\mathcal{X})$.

Fix such Ψ. The map $\hat{E}_\Psi$ is continuous, being the composition of $\hat{E}$, which is continuous by Lemma A.1, with the projection $v \mapsto (v + \Psi)/\|v + \Psi\|$, which is continuous away from $-\Psi$. Since $\mathcal{M}_{\leq T}(\mathcal{X})$ is compact by the same lemma, $\hat{E}_\Psi$ is a homeomorphism onto its compact image $\hat{\mathcal{Z}} \subset \mathbb{R}^m$, and $\phi := \hat{E} \circ \hat{E}_\Psi^{-1} : \hat{\mathcal{Z}} \to \mathbb{R}^m$ is continuous with $\phi(\hat{m}_t) = m_t$ for every history. Extending ϕ to a continuous map $\tilde{\phi} : \mathbb{R}^m \to \mathbb{R}^m$ by the Tietze extension theorem and setting $\pi'(s, \hat{z}) = \pi(s, \tilde{\phi}(\hat{z}))$ gives a continuous function with

$$\pi'(s_t, \hat{m}_t) = \pi(s_t, m_t) = \pi^*(s_t, p(c|x_{1:t})), \tag{19}$$

which proves the claim. The same argument applies to the optimal value function, so Corollary A.2 extends in the same way. □

A.5 Proof of Proposition 5.1

Proposition 5.1 *Let $\mathcal{L}$ be the full-episode loss of Algorithm 1, obtained with $S = S' = \{1, \dots, T\}$, and let ϕ denote all trainable parameters other than ψ. Let S and S' be independent and uniform over the size-k subsets of $\{1, \dots, T\}$, $1 \leq k \leq T$, and assume $\bar{z}_i = \mathrm{sg}(E_\psi(x_i))$ for all i, where* sg *is the stop-gradient operator. Then $\mathbb{E}[\nabla_\phi \hat{\mathcal{L}}_{S'}] = \nabla_\phi \mathcal{L}$ and $\mathbb{E}[\beta \nabla_\psi \hat{\mathcal{L}}_{S'}] = \nabla_\psi \mathcal{L}$ with $\beta = T/k$.*

Proof. Write $\ell_t = \ell_t^{\mathrm{RL}}$ for the loss of transition x_t in Algorithm 1. Under the assumption $\bar{z}_i = \mathrm{sg}(E_\psi(x_i))$, every z_i in Algorithm 1 has the value $E_\psi(x_i)$ whatever S is. The memories $m_{0:T}$, the targets, and each ℓ_t therefore take the same values as in full-episode training. S only decides which z_i carry gradient, and S' which steps enter the loss.

For ϕ, the gradient $\nabla_\phi \ell_t$ involves the embeddings only through their values, so it is the same for every S. Since $\Pr[t \in S'] = k/T$,

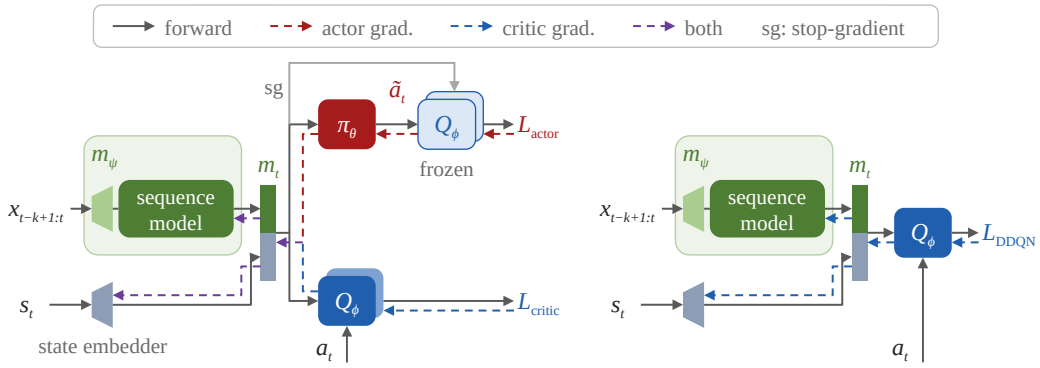
$$\mathbb{E}[\nabla_\phi \hat{\mathcal{L}}_{S'}] = \frac{1}{k} \sum_{t=1}^{T} \Pr[t \in S']\, \nabla_\phi \ell_t = \frac{1}{T} \sum_{t=1}^{T} \nabla_\phi \ell_t = \nabla_\phi \mathcal{L}.$$

For ψ, the next state enters ℓ_t only through the bootstrapped target, whose memory is held constant by sg. Hence ψ affects ℓ_t only through $m_{t-1} = \sum_{i<t} z_i$, in which every z_i with $i < t$ has coefficient one. When all embeddings carry gradient, the chain rule gives

$$\nabla_\psi \ell_t = \sum_{i<t} g_{t,i}, \qquad g_{t,i} = \Big(\frac{\partial E_\psi(x_i)}{\partial \psi}\Big)^\top \nabla_m \ell_t(m_{t-1}),$$

so $\nabla_\psi \mathcal{L} = \frac{1}{T} \sum_{t=1}^{T} \sum_{i<t} g_{t,i}$. In a STORE update the loss runs over $t \in S'$ and z_i carries gradient only if $i \in S$, so

$$\nabla_\psi \hat{\mathcal{L}}_{S'} = \frac{1}{k} \sum_{t \in S'} \sum_{i<t,\ i \in S} g_{t,i}$$



(a) SAC for continuous-action envs.

(b) DDQN for discrete-action envs.

Figure 5: **Memory-based RL framework used in our experiments.** The memory network m_ψ, a one-layer projection followed by the sequence model under comparison (MATE, GPT-2, LSTM, SplAgger, or Mamba), maps the last k transitions $x_{t-k+1:t}$ to the memory m_t, which is concatenated with an m-dimensional embedding of s_t. The twin critics Q_ϕ are trained on the replayed action a_t. The actor π_θ is trained through the critics evaluated at its own action $\tilde{a}_t$, with the critic parameters and the input (s_t, m_t) frozen. Dashed arrows trace the gradient of each loss. Both losses reach m_ψ and the state embedder (purple), so all networks are trained end-to-end by the RL loss alone.

with the same $g_{t,i}$. Since S and S' are independent, every pair satisfies $\Pr[t \in S' \text{ and } i \in S] = (k/T)^2$. Therefore

$$\mathbb{E}[\beta \nabla_\psi \hat{\mathcal{L}}_{S'}] = \frac{\beta}{k} \sum_{t=1}^{T} \sum_{i<t} \frac{k^2}{T^2} g_{t,i} = \frac{\beta k}{T} \nabla_\psi \mathcal{L} = \nabla_\psi \mathcal{L}$$

for $\beta = T/k$.

The argument uses only that ℓ_t depends on $z_{1:t-1}$ through their sum. The normalized memory of Appendix A.4 is a function of m_{t-1} and of $\Psi \in \phi$, so it can be regarded as the first layer of the networks, and the proposition holds unchanged with it. □

B Implementation Details

B.1 Memory-Based Agent and Baselines

Memory-based RL changes a standard off-policy algorithm in one place: the networks receive the memory m_t next to the state s_t. A memory network m_ψ maps the transition history $x_{1:t}$ to m_t, the actor and critic take (s_t, m_t) as input (Equation (2)), and all networks are trained jointly by the RL loss alone, with no auxiliary objective except in the Alchemy experiment of Section 6.3. Figure 5 shows the resulting agents for SAC and DDQN. In the input x_i the successor state s_i is replaced by the velocity $s_i - s_{i-1}$, which carries the same information, and states and transitions are standardized by moving-average feature normalization following AMAGO (Grigsby et al., 2024a). During rollout the memory is updated online from the full history as each transition arrives.

Training differs from the standard algorithm only in how a batch is formed. The replay buffer stores whole episodes, and each update samples a batch of episodes and, from each, a contiguous window of k transitions with a uniformly random start, where $k = T$ gives the whole episode. The memory network starts from its initial state at the first step of the window, so no information or gradient crosses the window boundary, which is the limit discussed in Section 5. The per-step losses of the base algorithm are averaged over the steps of the batch. MATE with STORE keeps this procedure and only replaces the window by the two random subsets of Algorithm 1.

Every architecture is matched to MATE in width m and in the number of layers n_{layers}: each stacks n_{layers} copies of the layer described below on top of a shared one-layer projection, a linear layer

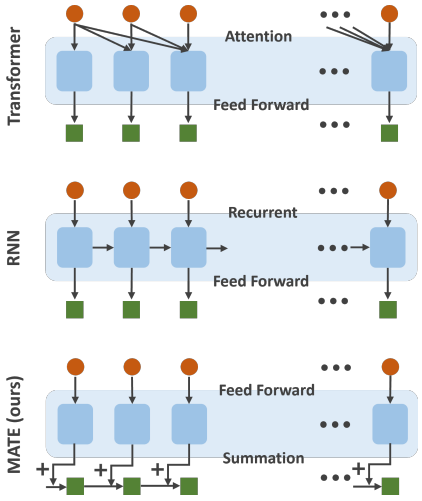


Figure 6: **Comparison of memory architectures.** MATE replaces the attention mechanisms of Transformers and the recurrent connections of RNNs with simple sum aggregation. Single-layer versions are illustrated for clarity.

Table 3: **Architecture and training settings per environment.** Memory dimension m and number of layers n_{layers}, shared by the memory architecture and the state embedder (the Markovian policy replaces the memory with a $2m$-wide state embedder), the number of transitions k per episode in each update, and the training budget in episodes, with the corresponding environment steps (episodes $\times T$, nominal for Meta-World, whose episodes can end early) in parentheses.

Environment	m	n_{layers}	k	Budget
Cheetah-Vel	256	1	32	50k (10M)
Ant-Dir	512	1	32	250k (50M)
Hopper-Param	256	2†	32	100k (20M)
Walker-Param	512	2†	32	250k (50M)
ML10	256	2	128	200k (100M)
ML45	512	2	128	200k (100M)
T-Maze	128	1	T	100k (10^5 T)
Vehicle Racing‡	256	1	32	40k (8M)
Alchemy	128	1	T	400k (80M)

†GPT-2 uses one layer on these two environments, where two layers performed poorly. ‡Image observations, encoded by a CNN shared by all agents before the embedders.

with LeakyReLU, that maps each transition to $\mathbb{R}^m$. For MATE the layer is a feed-forward layer, so the projection and the stack form the transition encoder E_ψ, whose outputs are summed and normalized as in Appendix A.4. GPT-2 and LSTM follow the implementation of Ni et al. (2023). The GPT-2 layer is a HuggingFace GPT-2 block that attends over one token per transition, with as many attention heads as layers and a sinusoidal positional encoding of the absolute time step within the episode, also inside a training window. The LSTM layer is a standard LSTM layer with orthogonal initialization. SplAgger, adapted from the official implementation of Beck et al. (2024), stacks GRU layers and splits the final output in half along the channels: one half is kept in order, the other is aggregated over time by a running maximum, and the two halves are concatenated. The Mamba layer is the official Mamba layer (Gu & Dao, 2024) with state dimension 16, convolution width 4, and expansion factor 2, wrapped in the pre-LayerNorm residual block of the AMAGO codebase (Grigsby et al., 2024a), with a final LayerNorm after the stack. The Markovian policy has no memory network and uses a $2m$-wide state embedder. Dropout of 0.05 is applied to the transition projection and inside the sequence modules.

Table 3 lists the memory dimension m, the number of layers n_{layers}, and the number of transitions k per episode in each update for every environment.

B.2 RL Algorithm Details

Both algorithms share one training loop. The agent performs 0.1 gradient updates per environment step, each on a batch of 64 episodes sampled as in Appendix B.1, using a single Adam optimizer for the memory network and the RL networks, a linear learning-rate warm-up over the first 500 updates, and gradient clipping by global norm. The critics have Polyak-averaged target copies, whereas the memory network has none: the bootstrapped target uses the memory that the current network produces for the next step, without gradient, following AMAGO (Grigsby et al., 2024a).

For continuous actions we use SAC (Haarnoja et al., 2018) with twin critics and automatic temperature tuning toward the target entropy $-\dim(\mathcal{A})$. The TD target takes the minimum of the two target critics at an action sampled from a Polyak-averaged copy of the actor, and critic outputs and targets are normalized with PopArt (Hessel et al., 2019) as in AMAGO. The actor loss is the standard $\alpha \log \pi_\theta(a \mid s_t, m_t) - \min_i Q_{\phi_i}(s_t, m_t, a)$, but it is evaluated with the critic parameters and the memory m_t treated as constants, the *freeze critic* setting in Table 4, so the memory network

is trained by the critic loss alone. This avoids the interference between the two losses on a shared memory network (Ni et al., 2022; 2023). Hyperparameters are listed in Table 4.

For discrete actions we use double DQN (Van Hasselt et al., 2016): the online network selects the next action, the target network evaluates it, and the loss is the squared TD error. Exploration is ϵ-greedy, with ϵ annealed linearly from 1 to $1/T$ over the first 10% of training and held there afterwards, following Ni et al. (2023). PopArt is not used on T-Maze. Hyperparameters are listed in Table 5.

Table 4: **SAC hyperparameters**

Hyperparameter	Value
Network hidden size	(512, 512)
Discount factor (γ)	0.99
Target update rate (τ)	0.001
Replay buffer size	1e4
Learning rate	1e-4
Gradient clipping	0.1
Freeze critic	true

Table 5: **DDQN hyperparameters**

Hyperparameter	Value
Network hidden size	(256, 256)
Discount factor (γ)	0.99
Target update rate (τ)	0.001
Replay buffer size	1e4
Learning rate	3e-5
Gradient clipping	0.03

B.3 Compute-Cost Measurement

We report two kinds of cost. Table 2 and the bottom row of Figure 3 give the wall-clock training time of complete RL runs, and Figure 4 isolates the cost of the memory module itself.

Training time is the wall-clock time of one run on a single NVIDIA V100-SXM2 (32 GB) GPU, including evaluation and logging, taken from the logged run durations. Preempted runs are timed per segment and the segments are summed, so a resumed run is charged the same time as an uninterrupted one. We report the median over seeds. Because the jobs ran on a shared cluster, seeds of the same configuration differ by up to about 25% on MuJoCo, where the environment steps on the CPU, and by a few percent on Meta-World, which is GPU-bound. In Table 2 we match n_{layers} across models within each environment, so GPT-2 is timed with two layers on Hopper-Param and Walker-Param.

Mamba and MATE with STORE were trained on different compute resources from the other models, so they have no V100 runs of their own. We estimate their training time from calibration runs on a rented 8×V100-SXM2 node. In each case we train the model and a reference model with identical settings for two thousand episodes, and the estimate is the ratio of their training rates multiplied by the V100 median of the reference. For Mamba we take the geometric mean over two references, LSTM and GPT-2. The same procedure predicts one baseline from another to within about 20%. For MATE with STORE the reference is MATE with the same k, which isolates the cost of STORE.

Figure 4 times the memory module alone, that is, the transition projection and the sequence module of Appendix B.1. Inputs use the dimensions of T-Maze, with $m = 128$, one layer, and a batch of 64 episodes. All modules are compiled with `torch.compile` and use TF32 matrix multiplication. Rollout is L sequential single-step calls without gradients, with GPT-2 caching its keys and values. Update is one forward and backward pass over the batch of length-L sequences with the dummy loss $\|m_t\|^2$. Time is the wall-clock time between device synchronizations and memory is the peak allocation of the call. We report the mean and standard deviation over 30 timed runs after 10 warm-up runs, measured on one V100-SXM2 per model with the process pinned to 10 CPU cores.

B.4 Compatibility with auxiliary objectives.

To enhance the learning signal, we adopt an auxiliary objective that maximizes an estimate of the mutual information between the memory and the context, a symmetric InfoNCE (Oord et al., 2018) over two statistics of the same episode,

$$\ell_k = \tfrac{1}{2}\Big[-\log \frac{e^{u_k^\top v_k/\tau}}{\sum_j e^{u_k^\top v_j/\tau}} - \log \frac{e^{u_k^\top v_k/\tau}}{\sum_j e^{u_j^\top v_k/\tau}}\Big],$$

Table 6: **MuJoCo tasks.** Robot, action dimension, and the distribution of the context drawn at the start of each episode.

Task	Robot	$\dim \mathcal{A}$	Context c
Ant-Dir	Ant	8	target direction $\theta \sim U(0, 2\pi)$
Cheetah-Vel	HalfCheetah	6	target speed $v^\star \sim U(0, 3)$
Walker-Param	Walker2d	6	23 dynamics multipliers 1.5^u, $u \sim U(-3, 3)$
Hopper-Param	Hopper	3	13 dynamics multipliers 1.5^u, $u \sim U(-3, 3)$

where u_k and v_k are normalized projections of the two views of anchor k, τ is a temperature, and the negatives are the other episodes in the batch, one context draw each.

The estimate concerns c only if the two views share no other source of variation, and such views are directly available in MATE. Since the memory accumulates transition embeddings, the statistic over any contiguous block of an episode is a difference of partial sums the forward pass has already produced, so for an anchor t we take m_t and $m_T - m_t$, each averaged over the transition it accumulates, the offset Ψ canceling in the difference. The two are built from disjoint transitions and cost nothing beyond the accumulation itself. A recurrent or attention based memory has no such shortcut, since each view over a different subset of transitions must be produced by running the sequence model again over that subset, once per anchor.

Anchors are drawn from the later portion of each episode, several per episode, so a pair contrasts a memory built from few transitions against the evidence still to come, and anchors of the same episode are excluded from each other's negatives. Pairs are weighted by $(1 - p_k)^\gamma$ with $p_k = e^{-\ell_k}$ (Lin et al., 2017) so that pairs the model has already matched do not crowd out the rest. The resulting term is added as $\mathcal{L} = \mathcal{L}_{\mathrm{RL}} + \lambda \mathcal{L}_{\mathrm{aux}}$ and optimized in a single backward pass, with gradients reaching E_ψ and the projection head only, leaving Q_ϕ, and the memory it consumes unchanged. We use $\tau = 0.03$, $\gamma = 2$, $\lambda = 0.005$, 16 anchors per episode drawn from its last 60 percent, and projection dimension 128.

C Environment Details

For each environment we state the hidden context, the observation and action spaces, the reward and termination rule, and every change we made to the original environment. Episode lengths T and training budgets are listed in Table 3.

C.1 MuJoCo

The four MuJoCo tasks are the locomotion suite of UNICORN (Li et al., 2024), built on the Gymnasium MuJoCo environments (Towers et al., 2026) Ant, HalfCheetah, Walker2d, and Hopper. Ant-Dir and Cheetah-Vel are the target-direction and target-velocity tasks of Finn et al. (2017), and Walker-Param and Hopper-Param randomize the physical parameters of Walker2d and Hopper, as PEARL (Rakelly et al., 2019) does for Walker2d. The observation and the torque action are the standard ones of each environment, and the observation excludes the context. A context is drawn at the start of every episode and held fixed (Table 6). Episodes last $T = 200$ steps and never end early: we disable the unhealthy-state termination of Gymnasium so that all episodes have the same length.

Ant-Dir and Cheetah-Vel change the reward and keep the dynamics. Ant-Dir uses the Gymnasium Ant reward with the forward velocity replaced by the velocity component along the target direction $(\cos\theta, \sin\theta)$, and Cheetah-Vel rewards $-|v_x - v^\star|$, the negative gap between the forward velocity and the target speed, minus the control cost. Walker-Param and Hopper-Param keep the standard forward-locomotion reward and change the dynamics: every body mass, body inertia, joint damping, and contact friction coefficient of the robot, 23 parameters for Walker2d and 13 for Hopper, is multiplied by 1.5^u with $u \sim U(-3, 3)$, i.e., by a factor between about 0.3 and 3.4. We reimplemented the environments of the UNICORN codebase for the current MuJoCo and Gymnasium versions with two changes: Walker-Param randomizes all four parameter groups, where the original randomizes mass and friction only, and damping uses the same base 1.5 as the other groups.

C.2 META-WORLD

ML10 and ML45 (Yu et al., 2020) are the Meta-World benchmarks with 10 and 45 training tasks. We use the `v3` environments of the current Meta-World codebase, whose task sets differ slightly from the `v1` suite of the original paper, and only their training tasks. All tasks share one embodiment, a simulated 7-DoF Sawyer arm with a parallel gripper on a tabletop, a 39-dimensional state observation, and a 4-dimensional action in $[-1, 1]$ for the end-effector displacement and the gripper. The context is the task together with its instance: at the start of every episode one task is sampled uniformly and then one of its 50 pre-generated instances, which fixes the object and goal positions. Following the ML protocol, the goal position is zeroed in the observation and the task identity is not given, so the agent must infer both from interaction. Episodes last $T = 500$ steps, and success does not end an episode.

ML10 comprises reach, push, pick-place, door-open, drawer-close, button-press-topdown, peg-insert-side, window-open, sweep, and basketball. ML45 widens the set to 45 tasks that cover reaching and pushing, opening and closing doors, drawers, windows, and faucets, pressing buttons and handles, and multi-stage tasks such as assembly, hammering, and stick pulling. Several of them, for example push, push-back, and push-wall, differ only in the goal or in an obstacle that the masked observation does not reveal, so the agent must distinguish far more hypotheses from the same interaction.

C.3 T-MAZE

T-Maze (Ni et al., 2023) is a grid world in which a corridor of length L leads to a junction with one cell above and one below it, the two goals. The context is which of the two goals is rewarded, drawn uniformly per episode. The agent moves left, right, up, or down by one cell per step. In *Passive* T-Maze the agent starts at the left end of the corridor, the oracle cell, where the goal is shown; in *Active* T-Maze it starts one cell to the right of the oracle cell and must first step left to see it. Episodes last $T = L + 1$ (Passive) and $T = L + 3$ (Active) steps, the length of the optimal path, and end when the agent reaches the rewarded goal or after T steps. Reaching the rewarded goal gives a reward of 1. Every step in which the agent does not advance along the corridor costs a penalty scaled so that a policy that never advances returns -1. The optimal policy therefore returns about 1, and a policy that guesses at the junction, the best a memoryless agent can do, returns 0.5 in expectation.

Adaptation to the CMDP framework. We modified the implementation of Ni et al. (2023) in four places so that the state is fully observed and only the context is hidden, as a CMDP requires. First, the original observation consists of two indicators, the goal cue and a junction flag, and all corridor cells look alike, so the agent must also infer its position; we observe the coordinates of the agent, normalized to $[-1, 1]$, together with a cue channel that equals the goal at the oracle cell and 0 elsewhere. Second, the original shows the cue only on the first visit to the oracle cell; we show it on every visit, so the observation is a fixed function of state and context. Third, the original penalty depends on the time step, charging the agent whenever it lags behind the position it would reach at full speed; ours charges each step without progress, so the reward depends on the transition alone. Fourth, in the Active variant we add a distractor and a constraint: at the first step the start cell shows a random cue that is independent of the goal, moving right is blocked, and no penalty is charged. Without them, an agent could read a cue at the start or skip the oracle cell, and the exploratory step would not be the only way to reveal the context. The original also runs every episode for T steps, whereas ours ends once the rewarded goal is reached.

C.4 VEHICLE RACING

Vehicle Racing is the Box2D *CarRacing* environment of Gymnasium (Towers et al., 2026), top-down driving around a closed track, with the vehicle-type context added by CARL (Benjamins et al., 2023). The context is the pair of the vehicle and the track, both sampled uniformly at the start of every episode. The vehicle is one of 29 classes, the combinations of four bodies (race car, street car, bus, and tuk-tuk), three drivetrains (rear-, front-, and all-wheel drive), and three trailer configurations (none, small, and large), which differ in engine power, wheel inertia, friction limits, mass distribution, steering lock, and, with a trailer, in the bodies and joints attached to the hull. The track is one of 10 fixed tracks: the original environment generates a new track on every reset, so that

transitions and rewards would vary between episodes for a reason outside the context, and we fix the set by reseeding the generator with a track index. Both components are only partially visible. The body and trailer of the vehicle and the road within the camera view are rendered, whereas the drivetrain and the course beyond the view are not.

The observation is a 96×96 RGB image from an ego-centric camera that follows and rotates with the car, with a strip at the bottom that shows the speed, the angular velocity of each wheel, the steering angle, the yaw rate, and the cumulative reward. We stack two consecutive frames into a $6 \times 96 \times 96$ observation, older frame first, because a single frame carries the speed only through the bar of the strip, quantized to about 20 speed units per pixel, and nothing about the direction of the velocity relative to the heading; two frames recover both by finite differences, and since the vehicle is a second-order system, position and velocity make the state Markov given the context. The action is 3-dimensional (steering, gas, and brake) in $[-1, 1]^3$, rescaled to the simulator's ranges, and each action is repeated for 8 simulator frames. An episode lasts $T = 200$ agent steps. The reward is that of CarRacing: -0.1 per simulator frame and $+1000/N$ the first time a wheel touches one of the N track tiles, with N between roughly 260 and 340, so that a lap yields 1000 minus the time penalty. An episode ends early if the car visits every tile or leaves the playfield, the latter with a penalty of -100. Rendering, reward, and termination are otherwise those of the original environment.

In this environment every agent encodes an observation with a CNN of two convolutional layers, with 32 and 64 channels, kernels of size 8 and 4, strides of 4 and 4, and ReLU activations, followed by a linear layer to 128 dimensions. The encoder is applied to the state and to both states of a transition before the state embedder and the transition projection of Appendix B.1, and is trained end to end with the rest of the agent.

C.5 ALCHEMY

We use the symbolic version of Alchemy (Wang et al., 2021). The agent is given stones and potions; potions change a stone's hidden properties, and dropping a stone into the cauldron cashes it in for a reward that depends on those properties. The context is the chemistry, the rules that connect what the agent sees to what a potion does. It is drawn once per episode and never shown, so the agent must work it out by experiment. An episode has 10 trials of 20 steps ($T = 200$), and every trial starts with 3 fresh stones and 12 fresh potions.

A stone's hidden state is a corner of a cube, $z \in \{-1, 1\}^3$. Cashing it pays $R(z) = \sum_i z_i$, except at the all-positive corner, which pays 15 instead of 3, and removes the stone. A potion moves a stone one step along a single axis, $z \leftarrow z + 2p$ with $p \in \{\pm e^{(i)}\}_{i=1}^3$, or leaves it in place if the step would exit the cube; the potion is used up either way. The chemistry has two parts. The potion map assigns the six potion colors, which come in fixed pairs that move along the same axis in opposite directions, to the three axes, which leaves $3! \cdot 2^3$ possibilities. The stone map determines how the hidden state appears to the agent, as a reflection of each of the three axes, which leaves 2^3. We use the *no-bottleneck* setting without perceptual rotation, so every potion applies to every stone, and there are $3! \cdot 2^3 \cdot 2^3 = 384$ chemistries.

Observations are normalized to $[-1, 1]$. Each stone shows three coordinates in $\{-1, 1\}$, a value channel $(\sum_i z_i)/3$, and a flag for whether it is still on the table; each potion shows its color and a flag for whether it has been used; three further channels mark the first step of a trial and report the remaining fractions of the trial and of the episode. An action is a no-op, applying one of the 12 potions to one of the 3 stones, or cashing one of the 3 stones, so $|\mathcal{A}| = 40$. Given the context, the observation is Markov: the stone map is one-to-one on the cube corners, so the observation determines the hidden state, and the used flags and timing channels cover the remaining budget. The value channel is in one-to-one correspondence with $R(z)$, so an agent can tell a good stone from a bad one and cash accordingly without knowing the chemistry. What the chemistry adds is knowing which potion to reach for, and 20 steps per trial are too few to find that out by trial and error every time.

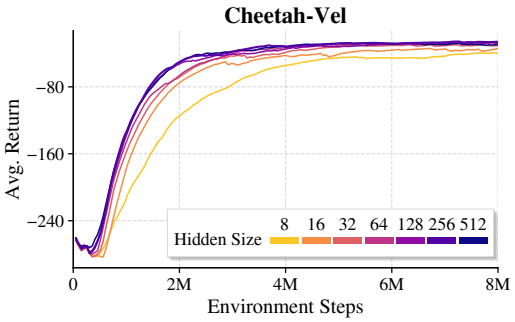


Figure 7: **Performance of MATE across memory dimensions.** Learning curves on Cheetah-Vel with memory dimensions ranging from 8 to 512. All configurations reach comparable final returns, indicating that end-to-end learning lets the encoder compress transition information into a compact low-dimensional representation. Consequently, the $2nT + 1$ dimensions that proposition 4.1 guarantees to be sufficient are far from necessary in practice.

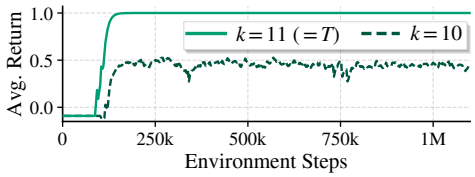


Figure 8: **Effect of truncated-window training on memory horizon.** Learning curves of GPT-2 on Passive T-Maze ($T = 11$) with window sizes $k = 10$ and $k = 11$. Since the task requires a context of length T to make the rewarding decision, $k = 10$ yields near-random performance while $k = 11$ attains the maximum return, indicating that the window size caps the effective memory horizon.

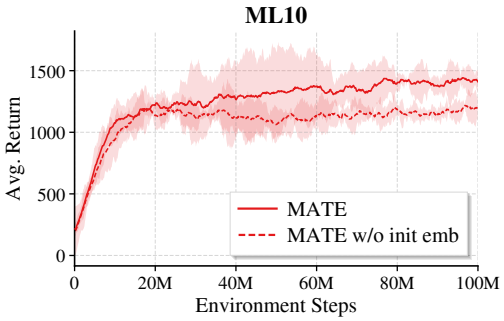


Figure 9: **Effect of the offset Ψ on ML10.** MATE with the learned offset (solid, reproduced from Figure 2) and with $\Psi = 0$ (dashed). Removing the offset degrades the final return.

D Additional Experiment Results

This appendix collects the additional results referenced in the main text: the memory dimension of MATE (Figure 7) and the pixel-observation experiment (Figure 11). Each caption states the setting and the result.

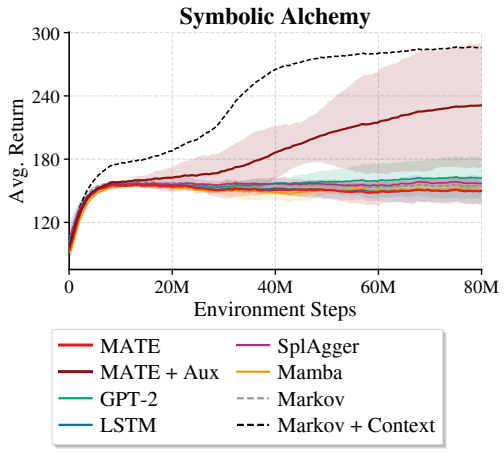


Figure 10: **Return Curves on Symbolic Alchemy** Results of Markov + Context is shown without its confidence band so that it stays distinguishable from Markov. Every memory based agent trained with the RL loss alone remains within a narrow band with the Markov policy. Only MATE with the auxiliary objective separates from that band, indicating that MATE's architecture makes it compatible with existing auxiliary obejctives.

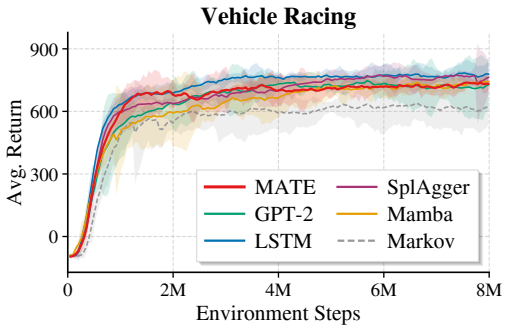


Figure 11: **Evaluation on a visual observation task.** Performance comparison on CARL Vehicle Racing, where observations are given as images. All agents share a CNN encoder in front of the transition and state embedders and are trained with SAC on the settings of Table 3; curves are the mean over 3 seeds. MATE matches the baselines in performance, indicating that it can learn effective memory even from high-dimensional visual inputs.