Memorax: Benchmarking Sequence Models for Memory in Reinforcement Learning

Anonymous authors
Paper under double-blind review

Keywords: reinforcement learning, POMDPs, sequence models, state space models, benchmarks

Summary

Memory is critical for solving partially observable tasks in reinforcement learning (RL), yet modern sequence models—state space models (SSMs), linear attention, and gated recurrent variants—have not been systematically evaluated in this setting. Existing POMDP benchmarks compare only a handful of memory architectures (typically GRU, LSTM, and Transformer) across a single algorithm. We introduce MEMORAX, an open-source JAX library providing 13 sequence models in a unified interface, supporting 5 RL algorithms across 15 POMDP environments, with additional support for multi-agent RL and intrinsic rewards. All models share a common `SequenceModel` interface; models amenable to parallel computation further implement the MEMOROID abstraction, which expresses their recurrence as an associative algebra for $O(\log n)$ training via prefix scan. We present a comprehensive benchmark comparing all 13 models with matched hidden state sizes ($\sim$512 elements) across PQN, PPO, and R2D2 on 15 POPJYM tasks, totaling 2,925 runs. Our results provide practical guidance on which memory architectures suit which problem types, and reveal that model rankings can shift substantially across algorithms.

Contribution(s)

1. MEMORAX: an open-source JAX library with 13 sequence models in a unified `SequenceModel` interface, supporting 5 RL algorithms (PPO, DQN, SAC, PQN, R2D2), multi-agent RL, and intrinsic rewards.
   Context: We provide a modular, JAX-native library that enables fair comparison of memory architectures for RL by standardizing the network architecture, training protocol, and evaluation across diverse sequence models and algorithms.
2. The MEMOROID abstraction: enables $O(\log n)$ parallel training of SSMs, linear recurrences, and linear attention via associative scan, unifying 8 architectures under a single algebraic interface.
   Context: By expressing parallelizable recurrent models as associative algebras, we enable efficient training via prefix scan while revealing structural similarities across seemingly different architectures.
3. A comprehensive benchmark: fair comparison of 13 models $\times$ 3 algorithms $\times$ 15 environments with matched hidden state sizes ($\sim$512 elements), totaling 2,925 runs with 5 seeds each.
   Context: This is the most comprehensive evaluation of sequence models for memory in RL to date, revealing that model rankings shift substantially across algorithms and providing practical guidance for practitioners.

Memorax: Benchmarking Sequence Models for Memory in Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

Memory is critical for solving partially observable tasks in reinforcement learning (RL), yet modern sequence models—state space models (SSMs), linear attention, and gated recurrent variants—have not been systematically evaluated in this setting. Existing POMDP benchmarks compare only a handful of memory architectures (typically GRU, LSTM, and Transformer) across a single algorithm. We introduce MEMORAX, an open-source JAX library providing 13 sequence models in a unified interface, supporting 5 RL algorithms across 15 POMDP environments, with additional support for multi-agent RL and intrinsic rewards. All models share a common `SequenceModel` interface; models amenable to parallel computation further implement the MEMOROID abstraction, which expresses their recurrence as an associative algebra for $O(\log n)$ training via prefix scan. We present a comprehensive benchmark comparing all 13 models with matched hidden state sizes ($\sim$512 elements) across PQN, PPO, and R2D2 on 15 POPJYM tasks, totaling 2,925 runs. Our results provide practical guidance on which memory architectures suit which problem types, and reveal that model rankings can shift substantially across algorithms.

1 Introduction

Many real-world reinforcement learning (RL) problems are partially observable: the agent must integrate information over time to infer the true state of the environment (Kaelbling et al., 1998). This memory requirement has traditionally been addressed by augmenting RL agents with recurrent neural networks such as LSTMs (Hochreiter & Schmidhuber, 1997) and GRUs (Cho et al., 2014), as in deep recurrent Q-learning (Hausknecht & Stone, 2015) and R2D2 (Kapturowski et al., 2019). More recently, Transformers (Vaswani et al., 2017) and their gated variants like GTrXL (Parisotto et al., 2020) have been applied to RL, offering richer memory through attention over past observations.

Meanwhile, the supervised learning community has witnessed a revolution in sequence modeling. State space models (SSMs) such as S4 (Gu et al., 2022), S5 (Smith et al., 2023), and Mamba (Gu & Dao, 2024) achieve strong performance on long-range tasks while scaling efficiently via parallel scan. Linear attention variants (Katharopoulos et al., 2020) and modernized gated recurrences like MinGRU (Feng et al., 2024), mLSTM (Beck et al., 2024), and Fast and Forgetful Memory (Morad et al., 2023b) further expand the design space. Despite their promise, these models remain underexplored in RL.

Existing POMDP benchmarks have not kept pace with this rapid development. POPGym (Morad et al., 2023a) introduced a diverse set of memory tasks but evaluated only 4 classical architectures (MLP, GRU, LSTM, Transformer). POBAX (Tao et al., 2025) added memory-improvability analysis but tested only 4 models with a single algorithm (PPO). Neither benchmark includes SSMs, linear attention, or modern gated recurrences, and neither examines how model rankings change across different RL algorithms.

We present MEMORAX, a JAX (Bradbury et al., 2018) library that addresses these gaps. MEMORAX provides 13 sequence models—spanning RNNs, SSMs, attention, and modern gated architectures—in a unified `SequenceModel` interface built on Flax (Heek et al., 2024). All models share a common network architecture (feature extractor $\rightarrow$ torso $\rightarrow$ head) and can be paired with any of 5 RL algorithms: PPO (Schulman et al., 2017), DQN (Mnih et al., 2015), SAC (Haarnoja et al., 2018), PQN (Gallici et al., 2025), and R2D2 (Kapturowski et al., 2019), with additional support for multi-agent RL and intrinsic rewards. Models amenable to parallel computation further implement the MEMOROID abstraction, which expresses 8 of the 13 models as associative algebras, enabling $O(\log n)$ parallel training via `jax.lax.associative_scan`.

We benchmark all 13 models across 15 POPJYM environments (Morad et al., 2023a) with 3 RL algorithms, using matched hidden state sizes ($\sim$512 elements) and standardized hyperparameters. The resulting 2,925 runs (with 5 seeds each) constitute, to our knowledge, the most comprehensive evaluation of sequence models for memory in RL to date.

2 Related Work

POMDP Benchmarks. POPGym (Morad et al., 2023a) introduced a suite of partially observable environments targeting specific memory capabilities including recall, retention, and pattern matching. The original benchmark evaluated GRU, LSTM, and Transformer architectures alongside several specialized memory modules, establishing baseline results across 15 task families. POPGym Arcade (Wang et al., 2025) extended this effort to pixel observations, requiring visual feature extraction before memory processing. POBAX (Tao et al., 2025) contributed a formal analysis of memory improvability—characterizing which environments genuinely benefit from memory—and evaluated 4 models (MLP, GRU, S5, Transformer) with PPO. Our work is complementary: we adopt the POPJYM (JAX-native) variant of POPGym environments and extend the evaluation from 4 to 13 models across 3 distinct RL algorithms.

Sequence Models in RL. The use of recurrent memory in RL has a long history, from LSTM-based deep Q-learning (Hausknecht & Stone, 2015) to R2D2's burn-in and stored-state techniques for stabilizing recurrent replay (Kapturowski et al., 2019). GTrXL (Parisotto et al., 2020) introduced gated Transformer-XL for RL, demonstrating improved stability over vanilla Transformers through gated identity map connections. Individual SSMs have been explored in RL—S5 and Mamba have shown promise on specific domains—but no prior work systematically compares the full range of modern SSMs (Mamba, LRU, S5), gated recurrences (MinGRU, sLSTM, mLSTM, SHM), and attention variants (self-attention, linear attention) in a controlled setting with multiple RL algorithms.

RL Libraries. CleanRL (Huang et al., 2022) provides clean single-file implementations of RL algorithms for reproducibility. PufferLib (Suarez, 2024) focuses on environment compatibility and vectorization speed. Stoix (Toledo, 2024) and PureJaxRL (Lu, 2023) offer JAX-native RL implementations emphasizing hardware acceleration. Syllabus (Sullivan et al., 2024) provides curriculum learning infrastructure for RL. None of these libraries focus on systematic memory architecture comparison. memax (Morad et al., 2025) provides a similar algebraic interface for sequence models in equinox, but does not include RL algorithms or benchmarking infrastructure. MEMORAX is the first library to combine a broad range of sequence models, multiple RL algorithms (including multi-agent and intrinsic reward variants), and a standardized benchmarking protocol in a single framework.

3 The MEMORAX Library

3.1 Architecture

MEMORAX follows a modular architecture where every agent's neural network is composed of three stages:

1. **Feature Extractor**: processes raw observations (and optionally previous actions, rewards, and done signals) into a fixed-size embedding vector. MEMORAX supports MLP, CNN, and Vision Transformer extractors, with optional embedding layers for discrete inputs.
2. **Torso**: the sequence model that maintains memory across time. The torso is composed as a `Stack` of interleaved `GatedResidual(PreNorm(SequenceModel))` and `GatedResidual(PreNorm(FFN))` blocks, following the architecture patterns established by modern language models. The gated residual connections (Beck et al., 2024) stabilize training and the pre-normalization allows for stable gradient flow through deep stacks.
3. **Head**: a task-specific output layer producing Q-values (`DiscreteQNetwork`), policy distributions (`Categorical`, `Gaussian`), value estimates (`VNetwork`), or distributional outputs (`C51QNetwork`).

This decomposition cleanly separates representation learning from temporal memory processing and task-specific outputs, enabling any sequence model to be combined with any RL algorithm through a shared `Network(feature_extractor, torso, head)` interface. In our benchmark, the feature extractor projects observations to 256 dimensions and embeds previous actions via a 32-dimensional embedding layer, concatenated and projected to the torso's input dimension.

3.2 Sequence Models

Table 1 lists all 13 sequence models implemented in MEMORAX, organized by architectural family. Each model implements a common `SequenceModel` interface with a uniform signature: given input features, an episode-boundary mask, and an optional hidden state (carry), the model returns the updated carry and a sequence of output features. This interface enables seamless swapping of any memory architecture into any RL algorithm. Models marked with $\star$ implement the MEMOROID interface (Section 3.3), enabling parallel training via associative scan.

The four architectural families are:

Traditional RNNs. GRU (Cho et al., 2014) and LSTM (Hochreiter & Schmidhuber, 1997) are wrapped via Flax's `nn.RNN` module and processed sequentially. sLSTM (Beck et al., 2024) extends LSTM with exponential gating and multi-head layer normalization. SHM (Le et al., 2025) uses Hadamard-product-based recurrence with learnable rotation matrices.

State Space Models. S5 (Smith et al., 2023) uses HIPPO-initialized diagonal state matrices with zero-order-hold discretization. LRU (Orvieto et al., 2023) parameterizes eigenvalues in polar form for stable complex-valued recurrence. Mamba (Gu & Dao, 2024) introduces input-dependent (selective) state transitions with a multi-head architecture.

Modern Gated Recurrences. MinGRU (Feng et al., 2024) reformulates GRU in log-space for numerical stability over long sequences. mLSTM (Beck et al., 2024) uses gated linear attention with a matrix-valued memory, combining key-value outer products with learned forget gates. FFM (Morad et al., 2023b) uses complex exponential basis functions with per-step decay for position-relative memory.

Attention. Self-Attention (Vaswani et al., 2017) operates over a fixed context window with RoPE positional embeddings and causal masking. Linear Attention (Katharopoulos et al., 2020) replaces softmax with an ELU+1 feature map, enabling linear-time recurrent computation.

3.3 The MEMOROID Abstraction

While all models in MEMORAX share the same `SequenceModel` interface, a subset of models—SSMs, linear recurrences, and linear attention—admit an additional optimization: their recurrence can be formulated as an *associative algebra* over carry states, enabling efficient parallel computation via prefix scan (Blelloch, 1990). The MEMOROID abstraction captures this structure.

Formally, a MEMOROID is defined by a tuple $(S, f, \oplus, \text{read})$ where:

Table 1: Sequence models in MEMORAX. **Family**: architectural category. **Computation**: sequential scan or parallel associative scan. **State size**: hidden state elements for our benchmark configuration ($\sim$512 target). Models marked $\star$ implement the MEMOROID interface.

Model	Family	Computation	Reference	State Size
GRU	RNN	Sequential	Cho et al. (2014)	512
LSTM	RNN	Sequential	Hochreiter & Schmidhuber (1997)	512
sLSTM	RNN	Sequential	Beck et al. (2024)	512
SHM	RNN	Sequential	Le et al. (2025)	529
S5 $\star$	SSM	Parallel	Smith et al. (2023)	512
LRU $\star$	SSM	Parallel	Orvieto et al. (2023)	512
Mamba $\star$	SSM	Parallel	Gu & Dao (2024)	512
MinGRU $\star$	Gated	Parallel	Feng et al. (2024)	256
mLSTM $\star$	Gated	Parallel	Beck et al. (2024)	544
FFM $\star$	Gated	Parallel	Morad et al. (2023b)	512
Self-Attention	Attention	Parallel	Vaswani et al. (2017)	—
Linear Attn. $\star$	Attention	Parallel	Katharopoulos et al. (2020)	512
MLP	None	—	—	0

- S is the set of carry states;
- $f : X \to S$ maps each input to a carry element;
- $\oplus : S \times S \to S$ is an associative binary operator;
- read : $S \times X \to Y$ maps accumulated state and input to output.

Given a sequence of inputs $x_1, \ldots, x_T$, the model first maps each input to a carry element $z_t = f(x_t)$, then computes the prefix scan:

$$h_t = z_1 \oplus z_2 \oplus \cdots \oplus z_t \tag{1}$$

using `jax.lax.associative_scan`, which executes in $O(\log T)$ parallel depth instead of $O(T)$ sequential steps. The output at each timestep is then $y_t = \text{read}(h_t, x_t)$.

Episode boundaries are handled by incorporating a reset signal into the binary operator: when a done flag is encountered at position t, the operator replaces the combined carry with the fresh element z_t, effectively restarting memory at episode boundaries without breaking the associative scan structure.

In code, each MEMOROID cell implements three methods:

- `__call__(x)` $\to$ carry: map input to algebra element $z_t = f(x_t)$
- `binary_operator(a, b)` $\to$ carry: the associative combination $a \oplus b$
- `read(h, x)` $\to$ output: extract output from accumulated state

Table 2 shows the specific algebras for each MEMOROID model. A striking pattern emerges: the SSMs (S5, LRU, Mamba) and FFM all share the *same* diagonal linear recurrence structure with combine rule $(\alpha_b \cdot \alpha_a,\ \alpha_b \cdot s_a + s_b)$, where α is a decay factor and s is the accumulated state. They differ only in how the decay and state are parameterized from the input. MinGRU uses the logaddexp operation for numerically stable log-space combination, while mLSTM uses a more complex 4-tuple with log-space stabilization for its matrix-valued memory. Linear Attention is the simplest algebra: pure additive accumulation of key-value outer products.

3.4 Fair Comparison Protocol

A meaningful benchmark requires controlled comparisons. We follow the protocol established by PopGym Arcade (Wang et al., 2025) and extend it to multiple algorithms:

Table 2: Algebraic formulations of MEMOROID cells. All SSMs share a diagonal linear recurrence; gated models and attention use distinct combination rules. Notation: $\text{sp}(\cdot) = \text{softplus}$, $\text{lse} = \text{logaddexp}$, $\phi(\cdot) = \text{ELU} + 1$.

Model	**Element** (z_t)	**Combine** $(a \oplus b)$
S5, LRU	$(\bar{\Lambda}, \bar{B}x_t)$	$(\alpha_b \cdot \alpha_a,\ \alpha_b \cdot s_a + s_b)$
Mamba	$(e^{\Delta_t A},\ B_t x_t \Delta_t)$	$(\alpha_b \cdot \alpha_a,\ \alpha_b \cdot s_a + s_b)$
FFM	$(e^{-\beta+i\omega},\ Bx_t)$	$(\alpha_b \cdot \alpha_a,\ \alpha_b \cdot s_a + s_b)$
MinGRU	$(\log z_t + \log \tilde{h}_t,\ -\text{sp}(z_t))$	$(\text{lse}(d_b + s_a, s_b),\ d_a + d_b)$
mLSTM	$(\log f_t, i_t k_t v_t^\top, i_t k_t, m_t)$	*decay-weighted accumulation*
Linear Attn.	$(\phi(k_t)v_t^\top,\ \phi(k_t))$	$(S_a + S_b,\ n_a + n_b)$

- **Matched hidden state size**: All models are configured to have approximately 512 hidden state elements (Table 1), controlling for memory capacity across architectures. For complex-valued models (S5, LRU, FFM), we count each complex number as 2 real elements.
- **Shared network architecture**: Every model uses the same feature extractor (single-layer MLP with 256 units plus a 32-dimensional action embedding), the same torso template (2 layers of GatedResidual(PreNorm(SequenceModel)) interleaved with GatedResidual(PreNorm(FFN))), and the same output head appropriate for each algorithm.
- **Standardized hyperparameters**: Learning rate 5×10^{-5} with linear decay to 0, gradient clipping at 0.5 (global norm), and Adam optimizer with $\epsilon = 10^{-5}$. These values follow the PopGym Arcade protocol.
- **Fixed training budget**: 20M environment timesteps per run.
- **Multiple seeds**: 5 independent random seeds per configuration.

All experiments use Hydra-based configuration management, ensuring exact reproducibility. The complete hyperparameter tables are provided in Appendix A.

4 Experimental Setup

4.1 Environments

We evaluate on 15 POPJYM environments (Morad et al., 2023a) spanning 5 task families, each with Easy, Medium, and Hard difficulty levels:

- **Autoencode**: The agent observes a sequence of symbols and must reproduce them in order after a delay. Tests sequential recall with increasing sequence lengths.
- **Count Recall**: The agent tracks occurrences of symbols and must report the count when queried. Tests counting and selective recall under increasing vocabulary sizes.
- **Repeat First**: The agent must remember and repeat the very first observation in an episode. Tests long-term retention as episode length increases.
- **Repeat Previous**: The agent must repeat the observation from k steps ago, where k varies by difficulty. Tests fixed-delay recall with increasing delay.
- **Concentration**: A card-matching game where the agent must remember card locations revealed during play. Tests associative memory with increasing board sizes.

Difficulty levels increase the sequence length, vocabulary size, or board size, placing progressively greater demands on memory capacity and precision. All environments are implemented in JAX, enabling full GPU vectorization across parallel environments without CPU-GPU synchronization overhead.

4.2 Algorithms

We evaluate three RL algorithms representing distinct paradigms for training recurrent agents:

- **PQN** (Gallici et al., 2025): Parallelized Q-Network, an on-policy value-based method that uses vectorized environments with TD(λ) targets and ϵ-greedy exploration. PQN avoids replay buffers entirely, making it the simplest algorithm in our comparison and the most directly comparable to prior PopGym benchmarks that use on-policy methods.
- **PPO** (Schulman et al., 2017): Proximal Policy Optimization, an on-policy policy gradient method with generalized advantage estimation (GAE; Schulman et al., 2016). PPO uses separate actor and critic networks that share the same torso architecture, providing a test of whether memory models can simultaneously support policy and value estimation.
- **R2D2** (Kapturowski et al., 2019): Recurrent Experience Replay in Distributed RL, an off-policy value-based method with prioritized sequence replay, burn-in, and stored hidden states. R2D2 tests whether sequence models maintain useful representations when trained on replayed, potentially off-distribution sequences—a challenge that disproportionately affects models with complex internal dynamics.

4.3 Models

We evaluate all 13 models from Table 1 plus an MLP baseline with no temporal memory, serving as a lower bound that indicates how much performance can be achieved from single-step observations alone. All recurrent models use the same torso architecture: 2 layers of GatedResidual(PreNorm(SequenceModel)) interleaved with GatedResidual(PreNorm(FFN)) blocks, with an expansion factor of 4 for the FFN and a total output dimension of 256. Hyperparameters are held constant across all models within each algorithm; we deliberately do not perform per-model tuning, as this better reflects the practical scenario where a practitioner selects a memory architecture without extensive architecture-specific search.

4.4 Protocol

Each of the 13 models is paired with each of the 3 algorithms across all 15 environments, yielding $13 \times 3 \times 15 = 585$ unique configurations. Each configuration is run with 5 independent random seeds, for a total of 2,925 training runs. For PQN and PPO, we use 128 parallel environments with 128-step rollouts. For R2D2, we use 128 parallel environments feeding into a prioritized replay buffer of 1M transitions, with 32-step burn-in and 128-step training sequences. All runs train for 20M environment timesteps. Results are logged to Weights & Biases for tracking and analysis.

4.5 Metrics

Following the recommendations of Agarwal et al. (2021) for reliable evaluation of RL algorithms, we report:

- **Mean episodic return**: the primary performance measure, averaged over the final 10% of training timesteps across 5 seeds. This captures final performance after convergence.
- **Interquartile mean (IQM)**: the mean of the middle 50% of runs, providing a robust aggregate measure that is less sensitive to outlier runs than the mean.
- **Performance profiles**: the empirical cumulative distribution of normalized scores across all tasks, enabling visual comparison of model robustness. A model whose curve is consistently above another's is better across more tasks.

Scores are min-max normalized per environment using the MLP baseline (no memory) as the lower bound, so a score of 0 indicates no benefit from memory and a score of 1 indicates the best performance observed for that environment.

5 Results

Results will be populated once experiments complete. The sections below describe the planned analyses.

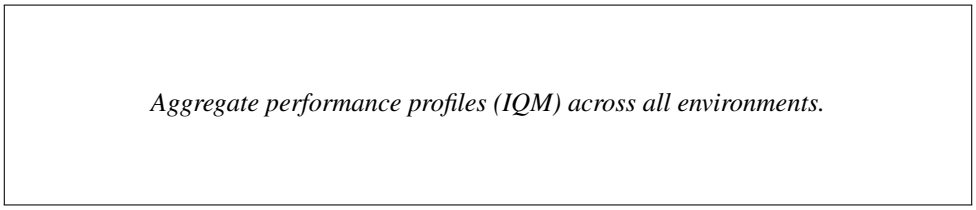

Figure 1: Aggregate performance profiles across all 15 POPJYM environments, comparing model families. Higher curves indicate better performance. Shaded regions show 95% stratified bootstrap confidence intervals.

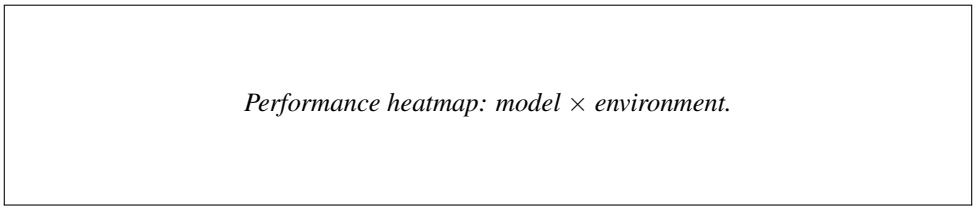

Figure 2: Normalized mean return for each model across all 15 environments (averaged over algorithms and seeds). Darker colors indicate higher performance. Models are sorted by aggregate IQM.

5.1 Aggregate Performance

We first examine aggregate performance across all 15 environments. Figure 1 shows the IQM performance profile for each model family, averaged across algorithms.

5.2 Per-Environment Analysis

Figure 2 presents a heatmap of normalized performance (model $\times$ environment), revealing which memory types suit which tasks.

5.3 Algorithm Sensitivity

A key question is whether model rankings are stable across RL algorithms, or whether the choice of algorithm confounds architecture comparisons. Figure 3 compares rankings under PQN, PPO, and R2D2 separately.

5.4 Computational Efficiency

We compare wall-clock training time for MEMOROID models (parallel scan) versus sequential RNN models to quantify the practical benefit of the associative scan formulation.

6 Discussion

Key Findings. *To be completed once results are available. We anticipate discussing:*

- Which model families dominate overall and in which task categories.
- Whether SSMs outperform classical RNNs, and whether attention still provides an advantage.
- The degree to which model rankings shift across PQN, PPO, and R2D2.
- Practical recommendations for practitioners choosing a memory architecture.

Limitations. Several limitations should be noted. First, POPJYM environments are relatively simple compared to real-world POMDPs—they isolate specific memory capabilities (recall, retention,

Model rankings across PQN, PPO, and R2D2.

Figure 3: Model rankings disaggregated by algorithm. Left: PQN. Center: PPO. Right: R2D2. Rank changes across algorithms indicate algorithm-sensitive models.

Wall-clock time comparison.

Figure 4: Steps per second (SPS) for each model during PQN training, measured on a single GPU. MEMOROID models (parallel scan) are compared against sequential RNN baselines.

pattern matching) rather than testing complex multi-step reasoning or long-horizon planning. Second, hidden state size matching is approximate: while all models target $\sim$512 state elements, the effective information capacity may differ due to architectural features. Complex-valued states (S5, LRU, FFM) encode phase information that has no real-valued analog, and structured states (mLSTM's matrix memory) may have different representational capacity per element than flat vectors. Third, hyperparameters are standardized rather than individually tuned per model, which may disadvantage architectures that are particularly sensitive to learning rate or other settings. Finally, we evaluate only discrete-action environments; continuous control POMDPs may yield different rankings, especially for policy gradient methods where the policy parameterization interacts with the memory architecture.

Future Work. Natural extensions include PopGym Arcade environments (pixel observations requiring CNN or ViT feature extractors), continuous control POMDPs (e.g., partially observable MuJoCo via proprioceptive masking), and multi-agent POMDPs (where communication and coordination create additional memory demands). Ablation studies on hidden state size (256 vs. 512 vs. 1024) and the number of stacked layers would further illuminate the interaction between memory capacity and task difficulty. The MEMOROID abstraction could also be extended to incorporate chunked or segmented scan variants for improved memory efficiency in extremely long sequences.

7 Conclusion

We presented MEMORAX, an open-source JAX library for benchmarking sequence models in memory-dependent reinforcement learning. By providing 13 models in a unified `SequenceModel` interface, 5 RL algorithms with support for multi-agent RL and intrinsic rewards, and a standardized evaluation protocol across 15 POMDP environments, MEMORAX enables the most comprehensive comparison of memory architectures for RL to date. Our benchmark of 2,925 runs reveals *[key finding to be filled in]* and provides actionable guidance for practitioners selecting memory architectures. We release all code, configurations, and experiment logs to facilitate reproducibility and future research at `[anonymous URL]`.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://arxiv.org/abs/2108.13264.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xLSTM: Extended long short-term memory. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2405.04517.

Guy E. Blelloch. Prefix sums and their applications. Technical Report CMU-CS-90-190, School of Computer Science, Carnegie Mellon University, 1990. URL https://www.cs.cmu.edu/~guyb/papers/Ble93.pdf.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734, 2014. DOI: 10.3115/v1/D14-1179.

Leo Feng, Frederick Tung, Mohamed Osama Ahmed, Yoshua Bengio, and Hossein Hajimirsadeghi. Were RNNs all we needed? *arXiv preprint arXiv:2410.01201*, 2024. URL https://arxiv.org/abs/2410.01201.

Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Nicolaus Foerster, and Mario Martin. Simplifying deep temporal difference learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2407.04811.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2312.00752.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *The Tenth International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2111.00396.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1861–1870, 2018.

Matthew Hausknecht and Peter Stone. Deep recurrent Q-learning for partially observable MDPs. In *AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents*, 2015. URL https://arxiv.org/abs/1507.06527.

Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2024. URL http://github.com/google/flax.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. CleanRL: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL http://jmlr.org/papers/v23/21-1342.html.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998. DOI: 10.1016/S0004-3702(98)00023-X.

Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *The Seventh International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=r1lyTjAqYX.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, 2020. URL https://arxiv.org/abs/2006.16236.

Hung Le, Kien Do, Dung Nguyen, Sunil Gupta, and Svetha Venkatesh. Stable Hadamard memory: Revitalizing memory-augmented agents for reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.10132.

Chris Lu. PureJaxRL: Really fast end-to-end JAX RL implementations, 2023. URL https://github.com/luchris429/purejaxrl.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riesenhuber, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Steven Morad, Ryan Kortvelesy, Matteo Bettini, Stephan Liwicki, and Amanda Prorok. POPGym: Benchmarking partially observable reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=chDrutUTs0K.

Steven Morad, Ryan Kortvelesy, Stephan Liwicki, and Amanda Prorok. Reinforcement learning with fast and forgetful memory. In *Advances in Neural Information Processing Systems*, volume 36, 2023b. URL https://arxiv.org/abs/2310.04128.

Steven Morad, Edan Toledo, Ryan Kortvelesy, and Zhe He. Memax: Deep memory and sequence models in JAX, 2025. URL https://github.com/smorad/memax.

Antonio Orvieto, Samuel L. Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *Proceedings of the 40th International Conference on Machine Learning*, 2023. URL https://arxiv.org/abs/2303.06349.

Emilio Parisotto, H. Francis Song, Jack W. Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant M. Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, Matthew M. Botvinick, Nicolas Heess, and Raia Hadsell. Stabilizing transformers for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, 2020. URL https://arxiv.org/abs/1910.06764.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2016. URL https://arxiv.org/abs/1506.02438.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Jimmy T.H. Smith, Andrew Warrington, and Scott Linderman. Simplified state space layers for sequence modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Ai8Hw3AXqks.

Joseph Suarez. PufferLib: Making reinforcement learning libraries and environments play nice. *arXiv preprint arXiv:2406.12905*, 2024. URL https://arxiv.org/abs/2406.12905.

Ryan Sullivan, Ryan Pégoud, Ameen Ur Rehman, Xinchen Yang, Junyun Huang, Aayush Verma, Nistha Mitra, and John P. Dickerson. Syllabus: Portable curricula for reinforcement learning agents. *arXiv preprint arXiv:2411.11318*, 2024. URL https://arxiv.org/abs/2411.11318.

Ruo Yu Tao, Kaicheng Guo, Cameron Allen, and George Konidaris. Benchmarking partial observability in reinforcement learning with a suite of memory-improvable domains. *Reinforcement Learning Journal*, 2025. URL https://arxiv.org/abs/2508.00046.

Edan Toledo. Stoix: A research-friendly codebase for fast experimentation of single-agent reinforcement learning in JAX, 2024. URL https://github.com/EdanToledo/Stoix.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://arxiv.org/abs/1706.03762.

Zekang Wang, Zhe He, Borong Zhang, Edan Toledo, and Steven Morad. Investigating memory in RL with POPGym arcade. *arXiv preprint arXiv:2503.01450*, 2025. URL https://arxiv.org/abs/2503.01450.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Hyperparameter Tables

Tables 3–5 list the complete hyperparameters for each algorithm. These are held constant across all memory architectures.

Table 3: PQN hyperparameters.

Hyperparameter	Value
Number of environments	128
Rollout length	128 steps
Number of minibatches	32
Update epochs	4
Discount (γ)	0.99
TD(λ)	0.95
Learning rate	5×10^{-5} (linear decay to 0)
Optimizer	Adam ($\epsilon = 10^{-5}$)
Gradient clip (global norm)	0.5
Epsilon schedule	$1.0 \rightarrow 0.05$ over 25% of training
Total timesteps	20M

Table 4: PPO hyperparameters.

Hyperparameter	Value
Number of environments	128
Rollout length	128 steps
Number of minibatches	32
Update epochs	4
Discount (γ)	0.99
GAE λ	0.95
Clip coefficient	0.2
Value loss clipping	True
Entropy coefficient	0.01
Value function coefficient	0.5
Advantage normalization	True
Learning rate	5×10^{-5} (linear decay to 0)
Optimizer	Adam ($\epsilon = 10^{-5}$)
Gradient clip (global norm)	0.5
Total timesteps	20M

B Model Configuration Details

Table 6 shows the exact configuration parameters for each model and how the $\sim$512 hidden state size target is achieved. All models use `features=256` as their output dimension (input to the head), with internal dimensions adjusted to reach the target state size.

C Full Learning Curves

Complete learning curves for all model $\times$ algorithm $\times$ environment combinations will be included in the final version.

Table 5: R2D2 hyperparameters.

Hyperparameter	**Value**
Number of environments	128
Replay buffer size	1M transitions
Batch size	128 sequences
Burn-in length	32 steps
Training sequence length	128 steps
Discount (γ)	0.99
Target network update (τ)	0.005 (Polyak averaging)
Epsilon schedule	$1.0 \rightarrow 0.01$ over 25% of training
Priority exponent (α)	0.9
IS exponent (β)	$0.6 \rightarrow 1.0$ (linear anneal)
Learning starts	5,000 environment steps
Train frequency	every 4 environment steps
Learning rate	5×10^{-5} (linear decay to 0)
Optimizer	Adam
Gradient clip (global norm)	0.5
Total timesteps	20M

Table 6: Detailed configuration for each model, showing how the $\sim$512 hidden state size target is achieved. Complex-valued elements are counted as 2 real elements.

Model	**Key Parameters**	**State Size Calculation**
GRU	features=512	512
LSTM	features=256	$256 \times 2 = 512$ (cell + hidden)
sLSTM	features=256, hidden_dim=64	$256 + 256 = 512$
SHM	features=23, output_features=256	$23^2 = 529$
S5	features=256, state_size=128	$128 \times 2 = 256$ (complex) $= 512$ real
LRU	features=256, hidden_dim=128	$128 \times 2 = 256$ (complex) $= 512$ real
Mamba	features=256, heads=2, head_dim=16, state_dim=16	$2 \times 16 \times 16 = 512$
MinGRU	features=256	256
mLSTM	features=256, hidden_dim=32, heads=2	$2 \times 16 \times (16 + 1) + 2 = 546$
FFM	features=256, memory=16, context=16	$16 \times 16 \times 2 = 512$ real
Self-Attention	features=256, heads=4, ctx=128	256×128 (KV cache)
Linear Attn.	heads=2, head_dim=16	$2 \times 16 \times 16 + 2 \times 16 = 544$
MLP	features=256	0 (stateless)