META: MAMBA EPISODIC TEMPORAL ALIGNMENT FOR STABLE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Structured State Space Models (SSMs), such as Mamba, offer an efficient mechanism for modeling long-range dependencies and are well suited to partially observable reinforcement learning (RL). However, directly applying Mamba to on-policy RL introduces two forms of temporal instability. First, we identify Δ-drift, where the learned temporal scaling parameter Δ, which governs the effective memory duration of the model, gradually diverges during training and destabilizes useful temporal context. Second, we observe episodic contamination, where residual state information persists across episode boundaries and violates episodic independence. To address these issues, we introduce META (Mamba Episodic Temporal Alignment), a framework for stabilizing the temporal dynamics of Mamba in episodic RL. META combines two complementary mechanisms: Episodic Forgetting, which attenuates temporal traces at episode boundaries to reduce cross-episode interference, and Memory Rebalancing, which regulates the sensitivity of Δ through structured spectral reparameterization to counteract drift during training. Across eight partially observable Brax benchmarks, META improves training stability and sample efficiency over recurrent and SSM-based on-policy baselines in most settings, achieving up to more than $2\times$ higher final returns than the strongest prior SSM baseline, S5RL, on tasks where that baseline attains positive performance. These results highlight temporal stability as a central challenge for deploying selective state-space models in episodic reinforcement learning.

1 INTRODUCTION

Structured State Space Models (SSMs) have recently emerged as a compelling backbone for sequence modeling due to their ability to capture long-range dependencies with efficient, hardware-friendly computation. In contrast to attention-based models, SSMs provide constant-cost recurrent inference and typically enable linear-time training through parallel scan operations, making them attractive for real-time and long-horizon decision-making problems (Gu et al., 2022). These properties align naturally with reinforcement learning (RL), where agents must process temporally extended streams of observations and actions under non-stationary and delayed feedback.

As a result, Mamba has achieved state-of-the-art performance across a wide range of domains, including robotic control (Liu et al., 2024; Tsuji, 2024), large-scale language modeling (Waleffe et al., 2024; Wang et al., 2025), computer vision (Zhu et al., 2024), and autonomous driving (Huang et al., 2025; Chen et al., 2025). These successes highlight the expressive power of input-dependent state-space models, motivating their adoption in broader sequential decision-making settings.

However, when directly used as a recurrent backbone in episodic, on-policy RL, input-dependent SSMs such as Mamba exhibit severe temporal instabilities that do not arise in offline or imitation settings.

Many challenging RL problems are better described as partially observable Markov decision processes (POMDPs), where the agent must infer hidden state from its history rather than relying on a fully observed Markov state (Kaelbling et al., 1998). In such settings, memory is essential: recurrent policies based on LSTMs improve performance by integrating information over time, but they can struggle with long-horizon credit assignment and unstable optimization dynamics (Hochreiter & Schmidhuber, 1997; Pascanu et al., 2013). Transformers are expressive sequence models and have been used in RL, but direct deployment in online, on-policy control creates a severe practical

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.

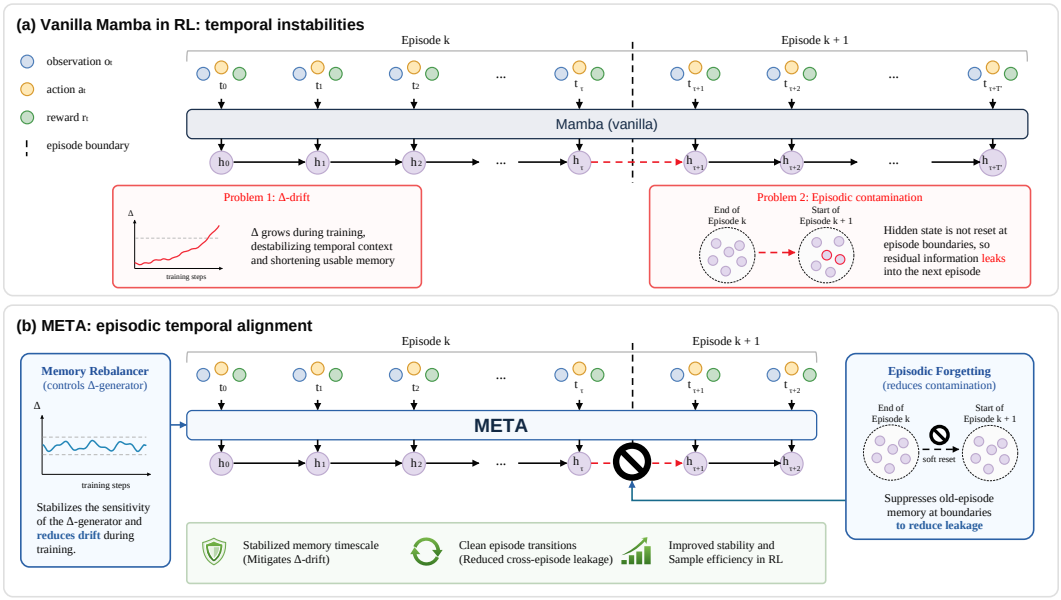

Figure 1: Overview of META for stabilizing Mamba-based episodic RL. A standard Mamba agent (top) suffers from two recurrent failure modes: Δ-drift, which shortens usable temporal context, and episodic contamination, in which residual hidden state carries across episode boundaries. META (bottom) addresses these failures by combining the Memory Rebalancer, which controls the Δ-generator's sensitivity, with Episodic Forgetting, which attenuates prior-episode memory at episode boundaries. Together, these mechanisms support stable temporal behavior and strategic actions.

mismatch: attention is costly at long horizons, and stable RL optimization often requires specialized architectural safeguards such as GTrXL-style gating (Chen et al., 2021; Parisotto et al., 2020). This leaves a gap between expressive sequence modeling and efficient, stateful control under partial observability.

Motivated by this gap, recent work has explored SSM-based alternatives for RL. S5RL (Lu et al., 2023) adapts structured SSMs by enabling efficient parallel hidden-state resets at episode boundaries, yielding a scalable approach to partially observable control. However, many such models rely on input-invariant dynamics, limiting context-dependent memory modulation. *Mamba*, a selective SSM with input-dependent state transitions, addresses this limitation and is particularly promising for RL, where the relevant time-scale varies with task phase, stochasticity, and observation ambiguity (Gu & Dao, 2023).

Despite this promise, deploying Mamba as a recurrent backbone for episodic *on-policy RL* introduces nontrivial stability challenges. Existing Mamba-based decision-making research has largely focused on offline RL and imitation learning, exemplified by Decision Mamba (Lv et al., 2024) and MTIL (Zhou et al., 2025), where persistent hidden-state propagation across episode boundaries is not required. These settings therefore mask the instability modes central to episodic on-policy RL, where agents must manage persistent hidden states across episodes while learning from nonstationary on-policy updates. In contrast, PPO repeatedly alternates between fresh trajectory collection and policy updates, making stable temporal dynamics, long-horizon memory, and episodic independence first-order requirements (Schulman et al., 2017).

In this work, we characterize two measurable forms of *temporal instability* when Mamba is used as a recurrent backbone for episodic on-policy RL. First, we observe Δ-*drift*, in which the learned temporal scaling parameter Δ gradually diverges during training. Δ controls the effective temporal scale of memory, and its drift toward values that aggressively discount historical context reduces useful temporal retention and induces myopic behavior, even in long-horizon tasks. Second, we measure *episodic contamination*, where residual internal state influences the policy after an independent

episode boundary. In the evaluated on-policy pipelines, these effects degrade useful temporal context and motivate direct interventions on boundary handling and temporal-scale generation.

A key challenge is that standard stabilization strategies do not directly apply. For RNN-based agents, episodic isolation is typically enforced via explicit hidden-state resets, and optimization issues are often mitigated through techniques such as gradient clipping or architectural gating (Pascanu et al., 2013). However, Mamba's efficiency stems from fused selective-scan operations and input-dependent transitions, making reset-aware modifications (as in input-invariant SSM approaches) difficult to integrate without compromising its computational advantages. Moreover, the instability we observe is driven not by a single fixed recurrent matrix, but by the unconstrained and input-sensitive evolution of Δ—a mechanism that behaves like a learned temporal gate and can be amplified under short-horizon RL objectives.

To address these issues, we propose **META** (Mamba Episodic Temporal Alignment), a structurally simple framework that regulates Mamba's temporal behavior while preserving its efficiency. META consists of two complementary mechanisms. **Episodic Forgetting** reduces cross-episode dependence by inducing a soft reset of temporal traces at episode boundaries without modifying Mamba's core scan kernel. **Memory Rebalancer** stabilizes the Δ-generator by controlling its sensitivity through structured spectral reparameterization, empirically reducing drift while maintaining input-adaptive temporal modulation. Together, these mechanisms stabilize input-dependent memory dynamics and support robust on-policy training of Mamba-based RL agents.

We validate META across eight partially observable Brax benchmarks, where it improves stability and sample efficiency over recurrent and SSM-based baselines in most settings. The same mechanisms may also extend to other segmented or discontinuous sequence settings where naive recurrence can lead to drift or cross-segment contamination. Our main contributions are:

- We directly measure Δ-*drift* and cross-episode recurrent-state contamination in input-dependent Mamba policies trained with episodic on-policy RL.
- We propose **META**, combining **Episodic Forgetting** and **Memory Rebalancer**, to structurally stabilize temporal dynamics without sacrificing Mamba's efficiency.
- In matched Mamba-2 PPO comparisons, META raises mean final return on seven of eight tasks; frozen EF boundary handling reduces eight-step old-history influence by $27.3\times$, and the visual cue policy retains perfect accuracy at a tenfold longer delay.

2 Background

2.1 Partially Observable Markov Decision Processes

Partially observable RL requires agents to act from histories rather than fully observed Markov states. A POMDP is defined by $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, \Omega)$, where observations $o_t \sim \Omega(\cdot \mid s_t)$ only partially reveal the latent state (Kaelbling et al., 1998). The agent must therefore compress its history $(o_{\leq t}, a_{<t})$ into an implicit belief or memory state sufficient for action selection, making temporal memory central to POMDP agents (Hausknecht & Stone, 2015).

2.2 Structured State Space Models

SSMs model sequential data using recurrent linear dynamics, enabling efficient long-range dependency modeling with linear computational complexity per step.

$$h_t = A\, h_{t-1} + B\, x_t, \quad y_t = C^\top h_t,$$

where h_t is the hidden state and x_t is the input. Under discretization, the effective transition and input matrices A and B depend on the step size parameter Δ, while C maps the hidden state to the output. Under a Zero-Order Hold (ZOH) discretization with step size Δ, the transition matrices become

$$A = \exp(\Delta\, \bar{A}), \quad B = \bar{A}^{-1}\big(\exp(\Delta\, \bar{A}) - I\big)\bar{B},$$

where $\bar{A}$ and $\bar{B}$ are the continuous-time system and input matrices. The parameter Δ modulates the temporal scale over which input signals influence the system state: small values allow for slow

information integration across longer time-scales, while larger values enable faster responsiveness to recent inputs.

Recent models such as Mamba (Gu & Dao, 2023; Dao & Gu, 2024) extend this framework by introducing input-conditioned parameters and semi-separable structures, achieving Transformer-level expressiveness with state-space efficiency. These properties have positioned SSMs, particularly Mamba, as promising backbones for sequence modeling in RL.

2.3 Temporal Modulation via Δ in Mamba Models

Mamba builds on a continuous-time linear state-space parameterization; after input-dependent selection and ZOH discretization with step size Δ_k, the resulting update can be written as:

$$h_{k+1} = e^{\Delta_k \bar{A}}\, h_k + \bar{A}^{-1}\big(e^{\Delta_k \bar{A}} - I\big)\bar{B}_k\, x_k,$$

where the continuous-time state parameter $\bar{A}$ is learned but input-independent, while $\bar{B}_k$, C_k, and Δ_k depend on x_k. Consequently, the discretized transition $A_k = e^{\Delta_k \bar{A}}$ is input-dependent. The parameter Δ_k controls how quickly the system evolves in time. When Δ_k is small, we can approximate the matrix exponential as $e^{\Delta_k \bar{A}} \approx I + \Delta_k \bar{A}$, yielding:

$$h_{k+1} \approx h_k + \Delta_k\, \bar{A}\, h_k + \cdots,$$

which allows the system to retain long-range dependencies.

On the other hand, suppose $\bar{A}$ is a stable matrix, meaning all its eigenvalues λ_i satisfy $\mathrm{Re}(\lambda_i) < 0$. Then, as $\Delta_k \to \infty$, we have $\lim_{\Delta_k \to \infty} e^{\Delta_k \bar{A}} = 0, \quad \lim_{\Delta_k \to \infty} \bar{A}^{-1}(e^{\Delta_k \bar{A}} - I) = -\bar{A}^{-1}$, which results in the update:

$$h_{k+1} \approx -\bar{A}^{-1}\bar{B}_k\, x_k,$$

indicating that the system becomes purely input-driven and discards its historical context. This illustrates that Δ serves as a learned temporal modulation mechanism, dynamically controlling the memory horizon of the model.

2.4 Temporal Stability in Sequence Models

Ensuring temporal stability and long-term information retention has long been studied in RNNs through orthogonal or unitary parametrizations, spectral normalization, gradient clipping, skip connections, and long-sequence recurrent designs (Arjovsky et al., 2016; Wisdom et al., 2016; Henaff et al., 2016; Miyato et al., 2018; Pascanu et al., 2013; Orvieto et al., 2023). Related work on recurrent memory and temporal gating also highlights the importance of controlling effective timescales (Tallec & Ollivier, 2018; Gu et al., 2020). These methods primarily target fixed or input-invariant recurrent dynamics. In contrast, Mamba makes B_k, C_k, and Δ_k input-dependent; the discretized transition $A_k = e^{\Delta_k \bar{A}}$ then varies with input even though $\bar{A}$ is learned and input-independent. The core instability we observe, Δ-drift, arises from the unconstrained input-dependent evolution of the temporal scaling parameter; therefore, stabilization must act on the mechanism that generates Δ rather than on $\bar{A}$ alone.

3 Method

3.1 Episodic Forgetting: Soft Reset via Temporal Scaling

Reinforcement learning environments are inherently episodic, where each trajectory constitutes an independent sample and must not influence subsequent episodes. However, modern on-policy RL pipelines process experience in batched rollouts for efficiency, often concatenating fragments from different episodes into a single sequence. Without explicit handling, this causes hidden states to propagate across episode boundaries, violating episodic independence.

For recurrent models, such as RNNs or GRUs, episodic isolation can be enforced through a hard reset of the hidden state ($h_t = 0$) at episode boundaries. State Space Models (SSMs), however, maintain memory through internal dynamical states that are not designed for direct intervention.

Input-independent SSMs such as S5RL address this by extending associative scan operators with reset-aware accumulation, using a binary termination signal to conditionally reinitialize the state.

Mamba, in contrast, derives its expressiveness from *input-dependent and time-varying* dynamics, where the effective state transition matrices depend on the input at each timestep. Although Mamba also relies on a highly optimized associative scan, its selective, input-conditioned recurrence makes S5RL-style hard resets impractical without modifying the fused selective-scan kernels. Introducing per-environment branching or reset logic into these kernels would significantly complicate implementation and degrade parallel efficiency. Rather than altering the kernel, we seek a reset mechanism that is structurally compatible with Mamba's existing design.

Soft reset via temporal scaling. We introduce a *soft reset* that suppresses previous SSM-state influence by setting the pre-softplus temporal-scale projection to D at episode boundaries. This intervention uses the existing fused Mamba-2 path during training.

The Mamba-2 state transition uses $\bar{A}_h = -\exp(A_{\log,h}) < 0$ per head. With learned temporal-scale bias b_h, the effective boundary scale is $\Delta_{h,\text{reset}} = \text{softplus}(D + b_h)$. For the deployed diagonal SSM state, the previous SSM-state contribution therefore contracts according to

$$\|\exp(\Delta_{\text{reset}}\bar{A})\|_2 = \exp\left(-\min_h \left[\exp(A_{\log,h})\,\text{softplus}(D + b_h)\right]\right) < 1.$$

We use $D = 5$ across the main experiments. Table 3 directly measures a $27.3\times$ reduction in the eight-step effect of swapped previous-episode history, while Table 8 measures the response over $D \in \{1, 3, 5, 7, 9\}$. The soft reset retains the fused convolution and selective-scan implementation; the exact segmented-reset control is evaluated separately.

3.2 Memory Rebalancer: Sensitivity Control for Adaptive Temporal Scaling

As shown in Figure 2, when trained without sensitivity control, the time-scale parameter Δ tends to grow uncontrollably. This phenomenon, which we term Δ-drift, causes the model to progressively discount historical context, lose its ability to use past observations, and eventually suffer a temporal collapse where the policy behaves like a simple feed-forward network. We observe this drift broadly across evaluated environments, suggesting it is a recurring failure mode for Mamba-based agents in on-policy RL.

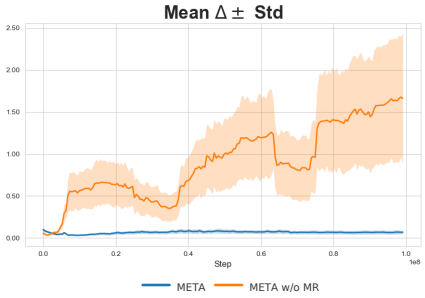

Figure 2: Δ (mean $\pm$ SD) over training steps. META reduces Δ-drift by controlling the sensitivity of the Δ-generator.

Mechanistic explanation of Δ-drift. We attribute this instability to a structural misalignment between Mamba's input-dependent dynamics and the on-policy optimization objective.

For a diagonal SSM with a stable state matrix $\bar{A}$, the discretized dynamics are $h_{t+1} = e^{\Delta\bar{A}}h_t + \bar{A}^{-1}(e^{\Delta\bar{A}} - I)\bar{B}x_t$. The immediate sensitivity of the hidden state to Δ is given by:

$$\frac{\partial h_{t+1}}{\partial \Delta} = e^{\Delta\bar{A}}\left(\bar{A}h_t + \bar{B}x_t\right).$$

Unrolling the recurrence shows that older inputs enter h_t through products of transition matrices. In contrast, on-policy RL provides credit through a much shorter effective horizon: GAE weights future terms by approximately $(\gamma_{\text{RL}}\lambda)^k$, and practical implementations additionally use *truncated BPTT* over a finite unroll length L. Consequently, information from $j < t - L$ can remain in h_t while receiving little direct credit assignment. Under this stable-diagonal approximation, these older contributions are attenuated roughly as $\exp(-\alpha\sum_i \Delta_i)$, so larger average Δ shortens the effective memory horizon. This can create an optimization pressure toward immediate-observation dependence, providing a mechanistic explanation for the observed Δ-drift.

Spectral Reparameterization of the Input Projection. To mitigate this optimization-induced instability, the Memory Rebalancer (MR) regulates the joint input projection W that produces z, x, B, C, and the Δ input. With an exact spectral norm $\sigma(W)$, the reparameterization is

$$\hat{W} = \frac{\gamma}{\sigma(W)} W$$

For this exact-norm form, $\|\hat{W}\|_2 = |\gamma|$. Since softplus is 1-Lipschitz, the Δ output slice $\hat{W}_\Delta$ obeys $\|\hat{W}_\Delta\|_2 \leq \|\hat{W}\|_2$. For a scalar channel (or componentwise for vector-valued Δ) with $\Delta(x) = \mathrm{softplus}(\hat{W}_\Delta x + b)$, this gives

$$\begin{aligned}\|\Delta(x) - \Delta(x')\|_2 &\leq \|\hat{W}_\Delta\|_2 \|x - x'\|_2 \\ &\leq \|\hat{W}\|_2 \|x - x'\|_2 = |\gamma| \|x - x'\|_2.\end{aligned}$$

The implementation tracks $\sigma(W)$ with power iteration, and the realized joint-projection gain is measured directly in Table 7. MR thereby controls an input projection that includes the temporal-scale pathway.

The same softplus parameterization ensures $\Delta_k > 0$: $\Delta_k = \mathrm{softplus}(\hat{W}_\Delta x_k + b)$. Furthermore, the state matrix $\bar{A}$ in Mamba is normal because it is diagonal. Let $\alpha = -\max_{\lambda \in \mathrm{spec}(\bar{A})} \mathrm{Re}(\lambda) > 0$ denote the stability margin.

For a normal matrix $\bar{A}$, the spectral norm of the state transition matrix $A_k = \exp(\Delta_k \bar{A})$ is bounded as follows:

$$\begin{aligned}\|A_k\|_2 &= \|\exp(\Delta_k \bar{A})\|_2 \\ &= \exp\left(\Delta_k \max_{\lambda \in \mathrm{spec}(\bar{A})} \Re(\lambda)\right) \\ &= e^{-\alpha \Delta_k} \leq 1, \quad (\text{since } \Delta_k > 0)\end{aligned}$$

This provides a mathematical sufficient condition under which the linear state-transition component is non-expansive, ruling out amplification through the A_k term itself.

The learnable parameter γ serves to regulate the sensitivity of Δ_k to the input during loss minimization, while the bias term b primarily controls the absolute offset of Δ_k.

This stability becomes especially critical in long-horizon tasks. In the long-horizon diagnostic in Figure 2, baseline models are vulnerable to temporal collapse, whereas META maintains a stable Δ range.

In conclusion, the Memory Rebalancer balances memory retention and adaptability by reparameterizing the joint input projection. Each training forward applies a small power-iteration update, adding matrix-vector work of order $\mathcal{O}(d_{\text{in}} d_{\text{out}})$ without changing Mamba's selective-scan kernel. In the evaluated architecture benchmark, META retains approximately 95.4–96.5% of the matched Mamba-2 training throughput (Table 13).

4 Experimental Results

4.1 Experimental Setup

We evaluate the META framework on diverse continuous control tasks from Brax (Freeman et al., 2021), modified into POMDP scenarios via custom observation masking. The Brax benchmarks use PPO. Beyond standard benchmarks, we extend our evaluation to POPGym tasks and long-horizon Brax stress tests with increased observation masking to assess robust memory retention.

Comparison Scope. We focus our comparison on recurrent and state-space architectures—specifically Mamba-1, LSTM, and S5RL—to prioritize models with explicit hidden-state propagation, while including GTrXL as a reference point. Furthermore, we conduct ablation studies using a simplified Mamba-2 PPO and variants without Episodic Forgetting (EF) or the Memory Rebalancer (MR) to isolate each component's contribution.

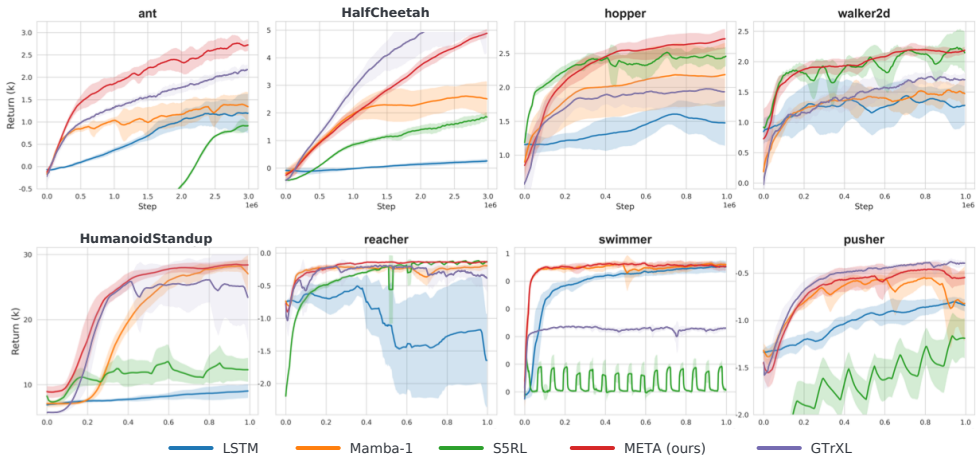

Figure 3: Performance comparison in eight Brax environments, showing cumulative returns over training steps (`Ant`, `HalfCheetah`, `Hopper`, `Walker2D`, `HumanoidStandup`, `Reacher`, `Swimmer`, and `Pusher`). The algorithms compared are LSTM, Mamba-1, S5RL, META (ours), and GTrXL. Shaded regions indicate standard deviations across multiple runs.

4.2 Benchmark Comparison

Task	Algorithm	AUC ($\times 10^6$)	Best Perf.	Final Perf.
HalfCheetah	S5RL	2823	1864	1770
	LSTM	173	258	237
	Mamba-1	5607	2605	2558
	GTrXL	**10480**	**5574**	**5253**
	META (ours)	7935	4882	4764
Ant	S5RL	−2054	916	863
	LSTM	1949	1204	1179
	Mamba-1	3081	1394	1377
	GTrXL	4764	2433	2293
	META (ours)	**5929**	**2765**	**2729**
Walker2D	S5RL	1890	**2235**	**2195**
	LSTM	1242	1437	1272
	Mamba-1	1293	1532	1504
	GTrXL	1545	2008	1843
	META (ours)	**1939**	2198	2159
Reacher	S5RL	−363	**−119**	−145
	LSTM	−1011	−505	−1268
	Mamba-1	−265	−195	−213
	GTrXL	−259	−130	−297
	META (ours)	**−201**	−133	**−144**

Task	Algorithm	AUC ($\times 10^6$)	Best Perf.	Final Perf.
Pusher	S5RL	−1739	−1164	−1223
	LSTM	−1022	−803	−819
	Mamba-1	−736	−549	−812
	GTrXL	**−572**	**−336**	**−392**
	META (ours)	−670	−459	−543
HumanoidStandup	S5RL	11342	13545	12368
	LSTM	7964	9032	8934
	Mamba-1	19460	28207	27934
	GTrXL	22407	27739	25817
	META (ours)	**23189**	**28494**	**28422**
Hopper	S5RL	2300	2518	2444
	LSTM	1358	1605	1486
	Mamba-1	1945	2188	2168
	GTrXL	1960	2257	2113
	META (ours)	**2308**	**2716**	**2684**
Swimmer	S5RL	28	126	35
	LSTM	392	454	452
	Mamba-1	442	464	**460**
	GTrXL	436	**478**	439
	META (ours)	**447**	465	454

Table 1: Benchmark metrics across Brax tasks. Bold indicates the best result; underlined entries denote the S5RL baseline. AUC is integrated return ($\times 10^6$), and Final is the mean over the last 10% of training.

Table 1 compares META with recurrent, SSM, and Transformer-based baselines across eight Brax tasks. META improves over the strongest prior SSM baseline (S5RL) in most settings, with gains up to 2.8× in AUC on `HalfCheetah`, and remains competitive with GTrXL while using a recurrent state-space backbone. It also reaches target performance faster in several tasks, including `HumanoidStandup`, `Pusher`, and `Ant`, supporting the view that EF and MR improve both stability and sample efficiency under partial observability.

4.3 Ablation Study

Task	META (EF+MR)	EF-only	MR-only	Vanilla Mamba-2
HalfCheetah	10932.3	**11245.5**	10371.4	9192.2
Ant	2024.0	1654.8	**3164.4**	538.6
Walker2D	457.9	531.3	**583.3**	458.6
Reacher	**−84.3**	−831.9	−339.8	−442.5
Pusher	−202.7	**−151.8**	−252.3	−196.6
HumanoidStandup	**28332.5**	25778.3	28233.1	23521.1
Hopper	**566.9**	551.8	536.3	510.9
Swimmer	40.2	41.5	**43.7**	24.1

Table 2: Architecture-matched Mamba-2 factorial: mean final return over 64 deterministic episodes (paired seeds 0–4). META uses Episodic Forgetting (EF) and Memory Rebalancer (MR); ablations remove either or both. All arms share the backbone, PPO budget, observation mask, and initialization. Bold marks the highest task mean.

Table 1 reports the cross-architecture comparison; Final averages return over the last 10% of training. Tables 2 and 6 report architecture-matched Mamba-2 experiments with explicit sequence-axis and episode-reset handling and isolated random streams. Their Final values average 64 deterministic evaluation episodes per run at the final checkpoint; the factorial uses five paired seeds and the matched comparison uses ten. We interpret performance differences within each experimental comparison. Table 2 isolates EF and MR with four paired arms across eight tasks. Full META exceeds vanilla Mamba-2 in mean final return on six tasks. MR-containing arms reduce joint input-projection drift across all eight tasks (Table 7). The ten-seed matched META–vanilla comparison improves mean final return on seven tasks; paired bootstrap intervals exclude zero on `Ant`, `Reacher`, and `Swimmer` (Table 6).

Boundary handling	Step 0	8-step mean
No reset	0.012810	0.004103
EF with $D = 5$	0.000176	0.000150
No-reset / EF ratio	72.8×	27.3×

Table 3: Frozen Swimmer history swap (seeds 0–2; 20,480 steps, 224 boundaries per seed). Entries are mean absolute logit changes.

At fixed policy weights and post-boundary observations, EF reduces the effect of swapped prior-episode history on policy logits. The step-0 change falls by 72.8×; the eight-step mean falls by 27.3× (Table 3). Both conditions use the same policies and boundary corpus, isolating the effect of boundary handling from retraining.

4.4 Controlled Long-Delay Visual Memory

We tested visual cue retention in a controlled 56×56 RGB Passive T-Maze. Each sequence packs two independent episodes; the policy observes an initial cue, traverses a neutral corridor, and acts at the final junction. Policies were trained at $T = 199$ and evaluated without further training at longer cue-to-decision delays.

Model	$T = 999$ (5×)	$T = 1999$ (10×)	$T = 2999$ (15×)	$T = 4999$ (25×)
META	100.0 ± 0.0	100.0 ± 0.0	82.3 ± 15.3	67.3 ± 28.3
EF-only	90.1 ± 17.1	85.3 ± 25.5	73.2 ± 26.4	64.2 ± 31.4
MR-only	49.8 ± 3.5	49.9 ± 1.4	48.4 ± 2.5	64.2 ± 25.5
Vanilla Mamba-2	61.7 ± 21.0	63.8 ± 20.9	64.1 ± 20.8	66.0 ± 29.4

Table 4: Frozen zero-shot accuracy (%) after training at $T = 199$ in the controlled RGB Passive T-Maze (paired seeds 0–2; mean ± SD). META retains 100% accuracy at 5× and 10× longer cue-to-decision delays.

Inverting only the initial visual cue changed META's final action in $100.0 \pm 0.0\%$ of cases at both 5× and 10× delay. Thus, its perfect accuracy at these horizons depends on the retained cue rather

than a fixed action preference. The action-flip rate fell to $66.7 \pm 28.9\%$ at $15\times$ and $33.3 \pm 57.7\%$ at $25\times$, consistent with the decline in zero-shot accuracy at longer delays.

4.5 Non-Visual Memory and Packed Sequence Boundaries

The non-visual Passive T-Maze isolates cue retention without visual extraction. Packed multi-query associative recall (MQAR) provides a sequence-model diagnostic with four independent segments.

Same-checkpoint diagnostic	EF on	EF off
Passive T-Maze, $L = 500$: META accuracy (%)	100.00 ± 0.00	48.96 ± 0.94
MQAR, META prefix-swap TV	0.0160 ± 0.0032	0.0707 ± 0.0476
MQAR, META prediction flips (%)	1.99 ± 0.73	9.64 ± 8.12
MQAR, EF-only prefix-swap TV	0.0191 ± 0.0205	0.0965 ± 0.0508
MQAR, EF-only prediction flips (%)	2.28 ± 2.01	11.91 ± 5.85

Table 5: Frozen EF on/off interventions at the same checkpoint (paired seeds 0–2; mean $\pm$ SD). TV denotes total variation of model predictions under a swapped preceding segment. The fixed-checkpoint comparison directly isolates boundary handling from training-time regularization.

4.6 Performance and Stability in Noisy Environments

To rigorously assess robustness, we selected two environments—`Hopper` and `Walker2D`—where the performance gap between methods is typically minimal. We injected Gaussian random noise with standard deviation 0.3 into the observation space to create challenging noisy control tasks (`noisy-Hopper`, `noisy-Walker2D`). We use this setting to probe both temporal robustness and the ability to maintain reliable sequence modeling in the presence of persistent observation noise.

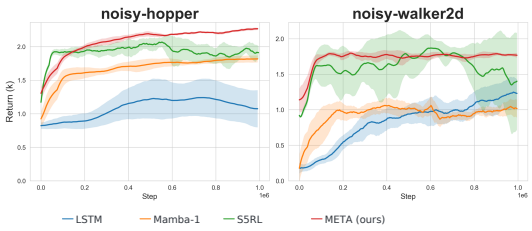

Figure 4: Noisy-control learning curves with Gaussian observation noise (SD $= 0.3$).

As illustrated in Figure 4, our proposed META achieves stronger returns than the tested baselines on noisy control tasks, such as `noisy-Hopper` and `noisy-Walker2D`. Notably, META not only achieves the highest average return, but also exhibits remarkably low variance across training runs—demonstrating both robust learning and reliable generalization in the presence of observation noise.

This high degree of stability is particularly evident in the narrow confidence intervals for META, contrasted with the significantly larger fluctuations observed in the baselines. The results suggest that explicit temporal stabilization—through Memory Rebalancer and Episodic Forgetting—effectively mitigates the compounding effects of noise, which commonly destabilize recurrent and SSM-based agents.

5 Conclusion

In this work, we introduced META, a framework designed to stabilize the temporal dynamics of Mamba-based SSMs for episodic RL. META addresses the critical instability issues—episodic contamination and uncontrolled temporal drift—that complicate direct use of Mamba in on-policy RL settings.

Across the matched ten-seed Mamba-2 comparison, META has higher mean final return on seven of eight partially observable control tasks, and the eight-task factorial links MR to lower projection drift. Frozen EF boundary handling reduces cross-episode logit sensitivity by $27.3\times$, while controlled visual and non-visual memory probes confirm retention and boundary isolation. The cross-architecture and noisy-control comparisons provide further evidence of stable training.

AI USE STATEMENT

Generative AI assisted with manuscript organization and editing, ICLR LaTeX preparation, construction of SVG figure assets, and tabulation and checking of author-recorded experiments. It was not used to devise the original method or hypotheses, design or run experiments, implement the original model, or generate experimental measurements. Numerical claims were checked against the recorded results, and the authors take responsibility for the final text, equations, figures, and tables.

REPRODUCIBILITY STATEMENT

Sections 3.1 and 3.2 specify the two mechanisms. Appendix B lists task-specific observation masks; Appendix C gives PPO settings, software, hardware, and the cross-architecture evaluation protocol. The matched Mamba-2 factorial and ten-seed comparison report their seed sets and deterministic evaluation rules in Tables 2 and 6. The frozen boundary and memory experiments specify their interventions and metrics in Tables 3, 4, and 5.

REFERENCES

Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary Evolution Recurrent Neural Networks, 2016. URL https://arxiv.org/abs/1511.06464.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 15084–15097, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/7f489f642a0ddb10272b5c31057f0663-Abstract.html.

Siran Chen, Yuxiao Luo, Yue Ma, Yu Qiao, and Yali Wang. H-MBA: Hierarchical MamBa adaptation for multi-modal video understanding in autonomous driving, 2025. URL https://arxiv.org/abs/2501.04302.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality, 2024. URL https://arxiv.org/abs/2405.21060.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax – a differentiable physics engine for large scale rigid body simulation, 2021. URL https://arxiv.org/abs/2106.13281.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2023. URL https://arxiv.org/abs/2312.00752.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. HiPPO: Recurrent memory with optimal polynomial projections. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1474–1487. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/102f0bb6efb3a6128a3c750dd16729be-Abstract.html.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=uYLFoz1vlAC.

Matthew J. Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable MDPs, 2015. URL https://arxiv.org/abs/1507.06527.

Mikael Henaff, Arthur Szlam, and Yann LeCun. Recurrent orthogonal networks and long-memory tasks. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 2034–2042. PMLR, 2016. URL https://proceedings.mlr.press/v48/henaff16.html.

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

Yizhou Huang, Yihua Cheng, and Kezhi Wang. Trajectory Mamba: Efficient attention-Mamba forecasting model based on selective SSM, 2025. URL https://arxiv.org/abs/2503.10898.

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.

Jiaming Liu, Mengzhen Liu, Zhenyu Wang, Lily Lee, Kaichen Zhou, Pengju An, Senqiao Yang, Renrui Zhang, Yandong Guo, and Shanghang Zhang. RoboMamba: Efficient vision-language-action model for robotic reasoning and manipulation, 2024. URL https://arxiv.org/abs/2406.04339.

Chris Lu, Yannick Schroecker, Albert Gu, Emilio Parisotto, Jakob Foerster, Satinder Singh, and Feryal Behbahani. Structured state space models for in-context reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://papers.nips.cc/paper_files/paper/2023/hash/92d3d2a9801211ca3693ccb2faa1316f-Abstract-Conference.html.

Qi Lv, Xiang Deng, Gongwei Chen, Michael Yu Wang, and Liqiang Nie. Decision mamba: A multi-grained state space model with self-evolution regularization for offline rl. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2406.05427. NeurIPS 2024.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks, 2018. URL https://arxiv.org/abs/1802.05957.

Steven Morad, Ryan Kortvelesy, Matteo Bettini, Stephan Liwicki, and Amanda Prorok. POPGym: Benchmarking partially observable reinforcement learning, 2023. URL https://arxiv.org/abs/2303.01859.

Antonio Orvieto, Samuel L. Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences, 2023. URL https://arxiv.org/abs/2303.06349.

Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, RL Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7487–7498. PMLR, 2020. URL https://proceedings.mlr.press/v119/parisotto20a.html.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training Recurrent Neural Networks, 2013. URL https://arxiv.org/abs/1211.5063.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Corentin Tallec and Yann Ollivier. Can recurrent neural networks warp time? In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SJcKhk-Ab.

Toshiaki Tsuji. Mamba as a motion encoder for robotic imitation learning, 2024. URL https://arxiv.org/abs/2409.02636.

Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak Narayanan, Garvit Kulshreshtha, Vartika Singh, Jared Casper, Jan Kautz, Mohammad Shoeybi, and Bryan Catanzaro. An empirical study of Mamba-based language models, 2024. URL https://arxiv.org/abs/2406.07887.

Junxiong Wang, Daniele Paliotta, Avner May, Alexander M. Rush, and Tri Dao. The Mamba in the Llama: Distilling and accelerating hybrid models, 2025. URL https://arxiv.org/abs/2408.15237.

Scott Wisdom, Thomas Powers, John R. Hershey, Jonathan Le Roux, and Les Atlas. Full-capacity Unitary Recurrent Neural Networks, 2016. URL https://arxiv.org/abs/1611.00035.

Yulin Zhou, Yuankai Lin, Fanzhe Peng, Jiahui Chen, Zhuang Zhou, Kaiji Huang, Hua Yang, and Zhouping Yin. MTIL: Encoding full history with mamba for temporal imitation learning. *arXiv preprint arXiv:2505.12410*, 2025. URL https://arxiv.org/abs/2505.12410.

Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision Mamba: Efficient visual representation learning with bidirectional state space model, 2024. URL https://arxiv.org/abs/2401.09417.

A Additional Experimental Evidence

A.1 Ten-Seed Matched Mamba-2 Comparison

The architecture-matched factorial (Table 2) and ten-seed comparison (Table 6) use task-specific PPO budgets and observation masks, explicit sequence-axis and episode-reset handling, isolated random streams, and deterministic evaluation. Final return averages 64 episodes per run, with rewards accumulated until termination.

Task	META (mean ± SD)	Vanilla (mean ± SD)	Paired Δ (95% CI)
HalfCheetah	11098.3 ± 3174.8	10620.8 ± 3062.2	$+477.4\ [-2779.2, 3405.1]$
Ant	2218.2 ± 1163.5	757.9 ± 893.5	$+1460.3\ [373.8, 2498.7]$
Walker2D	660.3 ± 647.0	527.4 ± 174.3	$+132.9\ [-158.2, 587.0]$
Reacher	-111.3 ± 74.7	-363.8 ± 311.8	$+252.4\ [70.0, 463.0]$
Pusher	-227.6 ± 118.5	-193.8 ± 75.7	$-33.9\ [-135.4, 50.4]$
HumanoidStandup	25484.2 ± 8183.4	24391.5 ± 8851.5	$+1092.7\ [-4191.6, 6958.0]$
Hopper	534.6 ± 75.6	495.5 ± 76.8	$+39.1\ [-6.5, 84.3]$
Swimmer	41.6 ± 11.3	28.7 ± 14.0	$+13.0\ [4.3, 21.9]$

Table 6: Final return for META and vanilla Mamba-2 in the ten-seed architecture-matched Brax PPO comparison. Each run's final score averages 64 deterministic evaluation episodes. The paired difference is META minus vanilla; intervals are percentile 95% bootstrap intervals over training seeds (20,000 resamples). All arms share task-specific PPO budgets, observation masks, and shared initialization.

A.2 Architecture-Matched Factorial Telemetry

Task	META (EF+MR)	EF-only	MR-only	Vanilla Mamba-2
HalfCheetah	0.136	10.464	0.161	11.031
Ant	0.178	4.117	0.174	4.413
Walker2D	0.064	1.569	0.046	1.564
Reacher	0.068	7.021	0.042	5.582
Pusher	0.035	0.940	0.026	1.139
HumanoidStandup	0.092	2.704	0.074	2.650
Hopper	0.013	0.041	0.010	0.072
Swimmer	0.046	13.022	0.076	11.703

Table 7: Layer-0 change in the effective norm of the joint input projection in the same eight-task Mamba-2 factorial (paired seeds 0–4). The projection generates z, x, B, C, and Δ; MR-containing arms show lower drift across the evaluated tasks.

The factorial return and projection-drift summaries use the same four Mamba-2 arms and paired seeds. MR-containing arms show substantially lower joint input-projection norm drift across all eight tasks. This projection generates the z, x, B, C, and Δ outputs, connecting MR to the temporal-scale generation pathway.

A.3 Frozen Episode-Boundary Interventions

We compare two evaluations of the same frozen policies and boundary corpus. Pre-boundary history is replaced by another trajectory, while policy weights and post-boundary observations remain fixed. This directly measures the effect of EF's boundary temporal-scale intervention on the policy's cross-episode sensitivity.

Table 3 reports the fixed-policy history-swap result in the main text.

The following dose response uses the same frozen Swimmer policies while varying only the boundary reset strength. It distinguishes influence from the preceding episode from influence of the new episode's first observation.

D	Old-history contamination ↓	New-cue influence at boundary	New-cue influence lags 1–7 ↑
1	1.1×10^{-3}	0.57 ± 0.08	0.02 ± 0.01
3	5.2×10^{-4}	0.54 ± 0.09	0.04 ± 0.01
5	1.5×10^{-4}	0.52 ± 0.09	0.06 ± 0.01
7	3.0×10^{-5}	0.51 ± 0.08	0.08 ± 0.01
9	1.2×10^{-5}	0.51 ± 0.08	0.09 ± 0.01

Table 8: Frozen-policy reset-strength sensitivity on Swimmer (mean $\pm$ SD, seeds 0–2 where shown). Old-history contamination is the eight-step mean absolute policy-logit change under a pre-boundary history swap. The other columns measure the response to the first observation of the new episode.

Table 9 reports paired final-return changes after retraining at $D = 1$ or $D = 9$ relative to $D = 5$.

Task	$D = 1$ minus $D = 5$	$D = 9$ minus $D = 5$
Ant	$+1050.1$	$+397.9$
Walker2D	$+25.5$	-31.0
Swimmer	-6.0	-2.0

Table 9: Mean paired final-return differences after retraining with each reset strength (seeds 0–4). Positive values favor the alternative reset strength over $D = 5$; the preferred value depends on the task.

In a separate comparison on Ant, HalfCheetah, and Swimmer (paired seeds 0–2), exact segmented reset showed no detectable final-return or AUC advantage over EF-only: all paired 95% intervals included zero. In a forced-boundary forward/backward microbenchmark, segmentation took $4.06\times$ as long as the unsegmented fused path.

A.4 Direct Temporal-Scale Clipping Control

Post-softplus clipping of Δ at 0.1 is a simpler way to cap temporal scale. Table 10 reports the matched three-task control; capping Δ did not uniformly reproduce the full method's final-return and AUC pattern.

	META		Direct clip $\Delta \leq 0.1$		Vanilla Mamba-2	
Task	Final	AUC	Final	AUC	Final	AUC
Ant	1937.4	634.7	1769.6	845.4	162.5	146.9
HalfCheetah	13547.9	8194.0	10519.2	5834.0	7492.1	6395.6
Swimmer	48.9	35.0	37.1	38.5	25.1	25.0

Table 10: Direct post-softplus Δ-clipping control under a matched three-task protocol (seeds 0–2). AUC is the endpoint-normalized training-curve metric within this protocol.

A.5 Episodic Forgetting Sensitivity and Boundary Contamination

This analysis tests whether Episodic Forgetting suppresses episodic contamination and whether the boundary pre-activation $D = 5$ is a robust default rather than a narrowly tuned value. We quantify contamination by swapping the pre-boundary history with a different trajectory and measuring the resulting mean absolute change in policy logits after the boundary. *Step-0 contamination* is this change at the first post-boundary step, while *window contamination* averages the same diagnostic over the displayed post-boundary window.

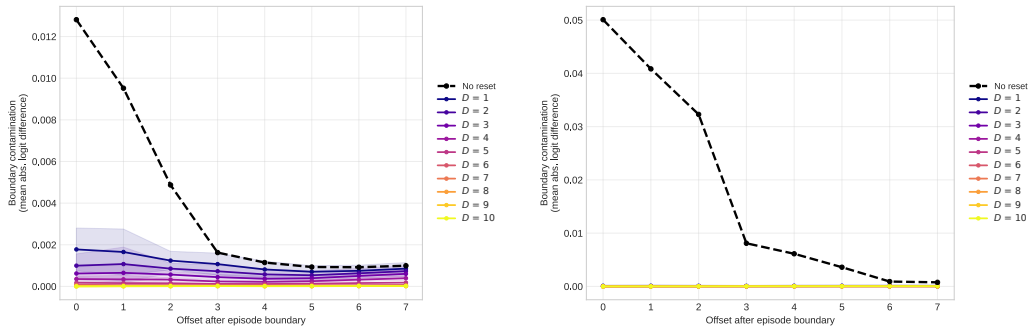

Figure 5: Episode-boundary contamination diagnostic for Swimmer (left) and Hopper (right). The diagnostic replaces pre-boundary history with another trajectory and measures post-boundary policy-logit change.

D	Score mean $\pm$ SD	Step-0 contamination	Window contamination
1	2.654 ± 0.425	0.004760 ± 0.005869	0.001581 ± 0.001432
3	2.582 ± 0.538	0.015848 ± 0.022010	0.003611 ± 0.004670
5	2.804 ± 0.147	0.000316 ± 0.000140	0.000239 ± 0.000069
7	2.510 ± 0.127	0.000042 ± 0.000037	0.000034 ± 0.000022

Table 11: Boundary pre-activation D sweep on Swimmer with return and boundary-contamination measurements.

A.6 POPGym Partial-Observability Experiments

POPGym provides an auxiliary check that the observed behavior is not limited to the custom masked-Brax wrapper. We treat these results as supporting evidence because performance varies across tasks. We do not use harder sparse-reward POPGym tasks as core evidence because the current PPO harness did not produce stable learning for any evaluated model on those settings. POPGym is a benchmark suite designed specifically for partially observable reinforcement learning (Morad et al., 2023).

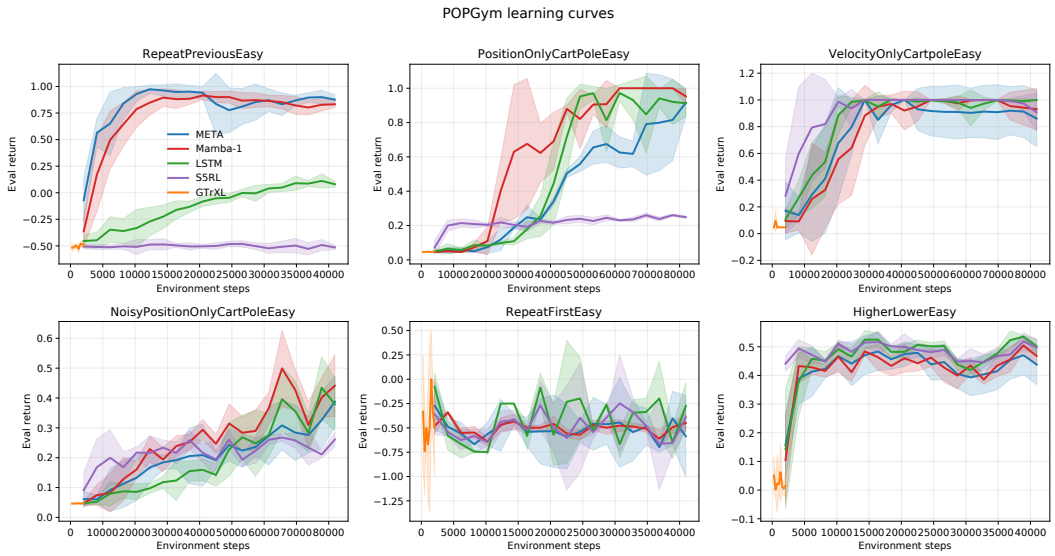

Figure 6: POPGym learning curves across six partially observable tasks.

A.7 Extended Brax Stress Tests

These stress tests evaluate longer episode horizons, stronger observation masking, and a linear-attention Transformer-style baseline under the same PPO recipe. They are intended as robustness

evidence rather than proof of extreme long-term memory. The *Linear attn.* model is the efficient Transformer-style comparator implemented as the `linear` model in the supplementary code. The mask labels are custom stress settings rather than Brax defaults: subset-A (*stronger*) keeps dimensions 0–8 for `Ant` and 0–5 for `HalfCheetah`, while subset-B (*much stronger*) keeps dimensions 0–4 and 0–2, respectively. Thus subset-B imposes the stricter partial-observability setting.

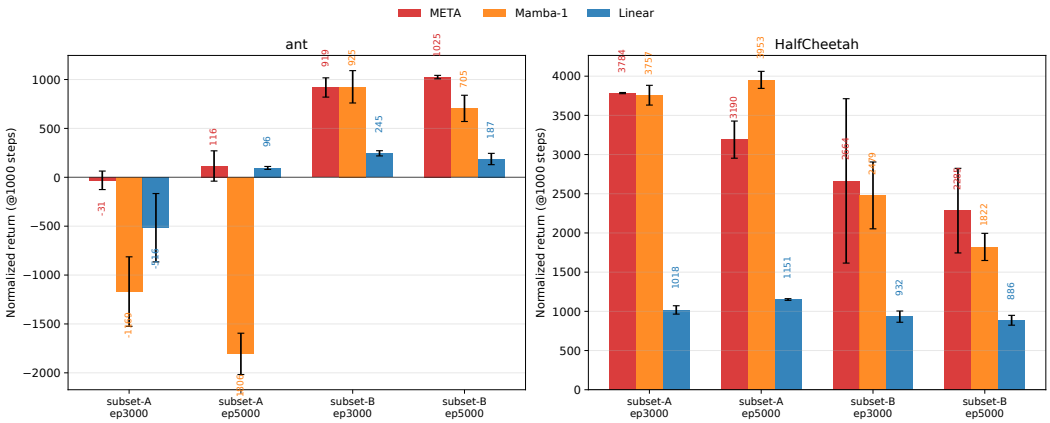

Figure 7: Brax stress-test results under longer horizons and stronger observation masking. Scores are normalized as raw return × 1000 / episode length; Linear denotes a linear-attention Transformer-style baseline.

Env	Mask	Horizon	META @1000	Mamba-1 @1000	Linear attn. @1000
Ant	subset-A (stronger)	ep3000	-31.04 ± 94.66	-1168.95 ± 355.46	-515.78 ± 349.55
Ant	subset-B (much stronger)	ep3000	918.59 ± 98.51	925.43 ± 165.44	245.21 ± 26.59
Ant	subset-A (stronger)	ep5000	115.55 ± 155.09	-1806.19 ± 211.33	95.65 ± 15.05
Ant	subset-B (much stronger)	ep5000	1024.83 ± 17.58	704.74 ± 134.23	186.92 ± 57.77
HalfCheetah	subset-A (stronger)	ep3000	3784.10 ± 7.53	3757.29 ± 125.98	1018.02 ± 53.46
HalfCheetah	subset-B (much stronger)	ep3000	2664.12 ± 1048.54	2479.14 ± 426.17	932.45 ± 71.79
HalfCheetah	subset-A (stronger)	ep5000	3190.43 ± 237.06	3952.75 ± 108.67	1151.18 ± 10.70
HalfCheetah	subset-B (much stronger)	ep5000	2284.73 ± 539.58	1821.87 ± 173.28	886.15 ± 62.78

Table 12: Integrated Brax stress-test summary. Scores are normalized as raw return × 1000 / episode length.

A.8 Computational Efficiency and Transformer Comparator

The stress-test results above include the *Linear attn.* baseline as an efficient Transformer-style comparator under the same PPO recipe. Table 13 further reports architecture-level timing for Mamba-2, META, and Linear attn. Training time measures one forward–backward update on a normalized synthetic rollout batch ($T = 128$, $B = 64$); step latency measures batched policy-forward latency, and FLOPs are profiler-visible policy-forward FLOPs.

Env	Model	Train Time (s)	Throughput (steps/s)	Rel. Throughput	Step Latency (ms)	Rel. Latency	FLOPs
Ant	Mamba-2	**0.018**	**467858.22**	**1.000x**	1.998	1.000x	8,290,308
Ant	META	0.018	451324.19	0.965x	1.989	0.996x	8,290,308
Ant	Linear attn.	0.267	30736.51	0.066x	**0.418**	**0.209x**	**632,833**
Walker2D	Mamba-2	**0.017**	**472288.79**	**1.000x**	2.037	1.000x	2,939,392
Walker2D	META	0.018	450781.60	0.954x	2.003	0.983x	2,939,392
Walker2D	Linear attn.	0.264	31010.72	0.066x	**0.415**	**0.204x**	**629,249**

Table 13: Representative architecture-level efficiency measurements. Relative throughput and latency are normalized to Mamba-2 within each environment. Bold indicates the best value within each environment for speed, latency, or profiler-visible FLOPs. Linear attn. denotes the linear-attention Transformer-style comparator used in Table 12.

B BRAX ENVIRONMENT

When constructing POMDP scenarios across various environments, we applied a consistent criterion for selecting observable dimensions. Specifically, absolute positional and angular values (e.g., meters or radians) were retained, as these provide critical information about the spatial and joint configurations of the agent. Rate-based quantities, such as velocities (e.g., meters per second), were excluded to simulate partial observability unless deemed essential for task-specific dynamics. These principles ensure a consistent methodology for POMDP construction while allowing controlled variations in observability to evaluate robustness and generalization across environments.

To evaluate the robustness of META in POMDP environments, we constructed POMDP scenarios within Brax, selecting a total of 8 environments (`Ant`, `Hopper`, `HumanoidStandup`, `Walker2D`, `HalfCheetah`, `Swimmer`, `Reacher`, and `Pusher`) for the experiment, with the elements used to configure the POMDP environment listed below.

Num	Observation	Name	Joint	Unit	Keep?
0	angle of the front tip	rot	hinge	angle (rad)	✓
1	angle of the second rotor	rot2	hinge	angle (rad)	✓
2	angle of the third rotor	rot3	hinge	angle (rad)	✓
3	velocity of the tip along the x-axis	slider1	slide	velocity (m/s)	✓
4	velocity of the tip along the y-axis	slider2	slide	velocity (m/s)	
5	angular velocity of front tip	rot	hinge	angular velocity (rad/s)	
6	angular velocity of second rotor	rot2	hinge	angular velocity (rad/s)	
7	angular velocity of third rotor	rot3	hinge	angular velocity (rad/s)	

Table 14: The `Swimmer` environment provides an 8-dimensional observation vector capturing the joint angles and velocities of a planar articulated agent. The task requires the agent to coordinate its body segments to swim forward efficiently in a low-friction environment. In our POMDP configuration, we retain only the joint angles and the velocity of the tip along the forward, while masking out all angular velocities and lateral motion. This selective observation design increases task ambiguity and challenges the agent to infer dynamics from minimal sensory feedback.

Num	Observation	Name	Joint	Unit	Keep?
0	Torso z-coord. ctr.	torso	free	pos. (m)	✓
1	Torso w-orient. ctr.	torso	free	unitless	✓
2	Torso x-orient. ctr.	torso	free	unitless	✓
3	Torso y-orient. ctr.	torso	free	unitless	✓
4	Torso z-orient. ctr.	torso	free	unitless	✓
5	Torso-FL link angle	hip_1	hinge	angle (rad)	✓
6	FL link1-2 angle	ankle_1	hinge	angle (rad)	✓
7	Torso-FR link angle	hip_2	hinge	angle (rad)	✓
8	FR link1-2 angle	ankle_2	hinge	angle (rad)	✓
9	Torso-BL link angle	hip_3	hinge	angle (rad)	✓
10	BL link1-2 angle	ankle_3	hinge	angle (rad)	✓
11	Torso-BR link angle	hip_4	hinge	angle (rad)	✓
12	BR link1-2 angle	ankle_4	hinge	angle (rad)	✓
13	Torso x-vel.	torso	free	vel. (m/s)	
14	Torso y-vel.	torso	free	vel. (m/s)	
15	Torso z-vel.	torso	free	vel. (m/s)	
16	Torso x-ang. vel.	torso	free	ang. vel. (rad/s)	
17	Torso y-ang. vel.	torso	free	ang. vel. (rad/s)	
18	Torso z-ang. vel.	torso	free	ang. vel. (rad/s)	
19	Torso-FL link ang. vel.	hip_1 (FL_leg)	hinge	ang. vel. (rad/s)	
20	FL links ang. vel.	ankle_1 (FL_leg)	hinge	ang. vel. (rad/s)	
21	Torso-FR link ang. vel.	hip_2 (FR_leg)	hinge	ang. vel. (rad/s)	
22	FR links ang. vel.	ankle_2 (FR_leg)	hinge	ang. vel. (rad/s)	
23	Torso-BL link ang. vel.	hip_3 (BL_leg)	hinge	ang. vel. (rad/s)	
24	BL links ang. vel.	ankle_3 (BL_leg)	hinge	ang. vel. (rad/s)	
25	Torso-BR link ang. vel.	hip_4 (BR_leg)	hinge	ang. vel. (rad/s)	
26	BR links ang. vel.	ankle_4 (BR_leg)	hinge	ang. vel. (rad/s)	

Table 15: The `Ant` environment provides a 27-dimensional observation vector, representing various physical properties of the agent. The observation vector includes information such as the position, orientation, and velocities of the torso and limb joints. The task involves controlling a quadrupedal robot to move forward efficiently while maintaining balance. Rewards are based on forward velocity, making coordination and stability critical under the partial observability imposed by the POMDP configuration.

Num	Observation	Name	Joint	Unit	Keep?
0	Top z-coord.	rootz	slide	pos. (m)	✓
1	Top angle	rooty	hinge	angle (rad)	✓
2	Thigh joint angle	thigh_joint	hinge	angle (rad)	✓
3	Leg joint angle	leg_joint	hinge	angle (rad)	✓
4	Foot joint angle	foot_joint	hinge	angle (rad)	✓
5	Top x-vel.	rootx	slide	vel. (m/s)	
6	Top z-vel.	rootz	slide	vel. (m/s)	
7	Top ang. vel.	rooty	hinge	ang. vel. (rad/s)	
8	Thigh hinge ang. vel.	thigh_joint	hinge	ang. vel. (rad/s)	
9	Leg hinge ang. vel.	leg_joint	hinge	ang. vel. (rad/s)	
10	Foot hinge ang. vel.	foot_joint	hinge	ang. vel. (rad/s)	

Table 16: The `Hopper` environment provides an 11-dimensional observation vector including torso and limb positions, joint angles, and velocities. The agent's goal is to maintain balance while hopping forward without falling. In our POMDP configuration, only absolute positional and angular features are retained (as indicated), while all velocity terms are masked out. This minimal set of observations increases the difficulty and encourages memory-based reasoning by the agent.

Num	Observation	Name	Joint	Unit	Keep?
0	Shoulder pan rot.	r_shoulder_pan_joint	hinge	angle (rad)	✓
1	Shoulder lift rot.	r_shoulder_lift_joint	hinge	angle (rad)	✓
2	Shoulder roll rot.	r_upper_arm_roll_joint	hinge	angle (rad)	✓
3	Elbow flex hinge rot.	r_elbow_flex_joint	hinge	angle (rad)	✓
4	Forearm roll hinge rot.	r_forearm_roll_joint	hinge	angle (rad)	✓
5	Wrist flex rot.	r_wrist_flex_joint	hinge	angle (rad)	✓
6	Wrist roll rot.	r_wrist_roll_joint	hinge	angle (rad)	✓
7	Shoulder pan ang. vel.	r_shoulder_pan_joint	hinge	ang. vel. (rad/s)	
8	Shoulder lift ang. vel.	r_shoulder_lift_joint	hinge	ang. vel. (rad/s)	
9	Shoulder roll ang. vel.	r_upper_arm_roll_joint	hinge	ang. vel. (rad/s)	
10	Elbow flex hinge ang. vel.	r_elbow_flex_joint	hinge	ang. vel. (rad/s)	
11	Forearm roll hinge ang. vel.	r_forearm_roll_joint	hinge	ang. vel. (rad/s)	
12	Wrist flex ang. vel.	r_wrist_flex_joint	hinge	ang. vel. (rad/s)	
13	Wrist roll ang. vel.	r_wrist_roll_joint	hinge	ang. vel. (rad/s)	
14	Fingertip x-coord.	tips_arm	slide	pos. (m)	✓
15	Fingertip y-coord.	tips_arm	slide	pos. (m)	✓
16	Fingertip z-coord.	tips_arm	slide	pos. (m)	✓
17	Object x-coord.	object_slidex	slide	pos. (m)	✓
18	Object y-coord.	object_slidey	slide	pos. (m)	✓
19	Object z-coord.	object	cylinder	pos. (m)	✓
20	Goal x-coord.	goal_slidex	slide	pos. (m)	
21	Goal y-coord.	goal_slidey	slide	pos. (m)	
22	Goal z-coord.	goal	sphere	pos. (m)	

Table 17: The `Pusher` environment provides a 23-dimensional observation vector describing the robot arm's joint positions and angular velocities, as well as the spatial coordinates of the fingertip, object, and goal. The task requires the agent to control a robotic manipulator to push an object toward a target location, demanding precise coordination under partial observability. In our POMDP configuration, only essential positional and angular features are retained to increase task difficulty and encourage inference-driven control.

Num	Observation	Name	Joint	Unit	Keep?
0	Torso z-position	rootz	slide	pos. (m)	✓
1	Torso pitch angle	rooty	hinge	angle (rad)	✓
2	Back thigh angle	bthigh	hinge	angle (rad)	✓
3	Back shin angle	bshin	hinge	angle (rad)	✓
4	Back foot angle	bfoot	hinge	angle (rad)	✓
5	Front thigh angle	fthigh	hinge	angle (rad)	✓
6	Front shin angle	fshin	hinge	angle (rad)	✓
7	Front foot angle	ffoot	hinge	angle (rad)	✓
8	Torso x-velocity	rootx	slide	vel. (m/s)	✓
9	Torso z-velocity	rootz	slide	vel. (m/s)	
10	Torso angular velocity	rooty	hinge	ang. vel. (rad/s)	
11	Back thigh angular velocity	bthigh	hinge	ang. vel. (rad/s)	
12	Back shin angular velocity	bshin	hinge	ang. vel. (rad/s)	
13	Back foot angular velocity	bfoot	hinge	ang. vel. (rad/s)	
14	Front thigh angular velocity	fthigh	hinge	ang. vel. (rad/s)	
15	Front shin angular velocity	fshin	hinge	ang. vel. (rad/s)	
16	Front foot angular velocity	ffoot	hinge	ang. vel. (rad/s)	

Table 18: The `HalfCheetah` environment provides a 17-dimensional observation vector: eight positional and angular components followed by nine velocity components. Our POMDP mask retains indices 0–8, including torso x-velocity at index 8, and removes the remaining velocity components. The agent's objective is to run forward as quickly and stably as possible.

Num	Observation	Name	Joint	Unit	Keep?
0	Top z-coord.	rootz (torso)	slide	pos. (m)	✓
1	Top angle	rooty (torso)	hinge	angle (rad)	
2	Thigh joint angle	thigh_joint	hinge	angle (rad)	✓
3	Leg joint angle	leg_joint	hinge	angle (rad)	✓
4	Foot joint angle	foot_joint	hinge	angle (rad)	✓
5	Left thigh joint angle	thigh_left_joint	hinge	angle (rad)	✓
6	Left leg joint angle	leg_left_joint	hinge	angle (rad)	✓
7	Left foot joint angle	foot_left_joint	hinge	angle (rad)	✓
8	Top x-vel.	rootx	slide	vel. (m/s)	
9	Top z-vel.	rootz	slide	vel. (m/s)	
10	Top ang. vel.	rooty	hinge	ang. vel. (rad/s)	
11	Thigh hinge ang. vel.	thigh_joint	hinge	ang. vel. (rad/s)	
12	Leg hinge ang. vel.	leg_joint	hinge	ang. vel. (rad/s)	
13	Foot hinge ang. vel.	foot_joint	hinge	ang. vel. (rad/s)	
14	Left thigh hinge ang. vel.	thigh_left_joint	hinge	ang. vel. (rad/s)	
15	Left leg hinge ang. vel.	leg_left_joint	hinge	ang. vel. (rad/s)	
16	Left foot hinge ang. vel.	foot_left_joint	hinge	ang. vel. (rad/s)	

Table 19: The `Walker2D` environment provides a 17-dimensional observation vector that includes the positions, joint angles, and velocities of the agent's torso and limbs. The agent's goal is to walk forward efficiently while maintaining upright posture and balance. To construct the POMDP setting, we retained only absolute positional and angular observations while masking out velocity-related features. This design encourages the agent to reason over time and rely on memory to infer movement dynamics.

Num	Observation	Name	Joint	Unit	Keep?
0	cosine of the angle of the first arm	cos(joint0)	hinge	unitless	✓
1	cosine of the angle of the second arm	cos(joint1)	hinge	unitless	✓
2	sine of the angle of the first arm	sin(joint0)	hinge	unitless	✓
3	sine of the angle of the second arm	sin(joint1)	hinge	unitless	✓
4	x-coordinate of the target	target_x	slide	position (m)	
5	y-coordinate of the target	target_y	slide	position (m)	
6	fingertip velocity along the x-axis	tip_vel_x	tip	velocity (m/s)	
7	fingertip velocity along the y-axis	tip_vel_y	tip	velocity (m/s)	
8	x-value of position_fingertip − position_target	NA	slide	position (m)	✓
9	y-value of position_fingertip − position_target	NA	slide	position (m)	✓
10	z-value of position_fingertip − position_target	NA	slide	position (m)	✓

Table 20: The `Reacher` environment provides an 11-dimensional observation vector representing trigonometric joint-angle encodings, target coordinates, fingertip velocities, and fingertip-target relative positions. The task requires a two-link robotic arm to reach a randomly placed target in 2D space using coordinated joint control. In our POMDP configuration, we retain only the sine and cosine of joint angles and the fingertip-to-target relative position, while masking out the target's absolute position and fingertip velocities. This setting significantly increases uncertainty and forces the agent to learn spatial reasoning purely from local and relative observations.

Num	Observation	Name	Joint	Unit	Keep?
0	Torso z-coord. ctr.	root	free	pos. (m)	✓
1	Torso w-orient. ctr.	root	free	angle (rad)	✓
2	Torso x-orient. ctr.	root	free	angle (rad)	✓
3	Torso y-orient. ctr.	root	free	angle (rad)	✓
4	Torso z-orient. ctr.	root	free	angle (rad)	✓
5	Abdomen z-angle lw.	abdomen_yz	hinge	angle (rad)	✓
6	Abdomen y-angle lw.	abdomen_yy	hinge	angle (rad)	✓
7	Abdomen x-angle pelvis	abdomen_x	hinge	angle (rad)	✓
8	Pelvis-R hip x-coord. angle	right_hip_xyz	hinge	angle (rad)	✓
9	Pelvis-R hip y-coord. angle	right_hip_xyz	hinge	angle (rad)	✓
10	Pelvis-R hip z-coord. angle	right_hip_xyz	hinge	angle (rad)	✓
11	R hip-shin angle knee	right_knee	hinge	angle (rad)	✓
12	Pelvis-L hip x-coord. angle	left_hip_xyz	hinge	angle (rad)	✓
13	Pelvis-L hip y-coord. angle	left_hip_xyz	hinge	angle (rad)	✓
14	Pelvis-L hip z-coord. angle	left_hip_xyz	hinge	angle (rad)	✓
15	L hip-shin angle knee	left_knee	hinge	angle (rad)	✓
16	Torso-R arm1 angle (RA-upper_arm)	right_shoulder12	hinge	angle (rad)	
17	Torso-R arm2 angle (RA-upper_arm)	right_shoulder12	hinge	angle (rad)	
18	R upper-lower arm angle	right_elbow	hinge	angle (rad)	
19	Torso-L arm1 angle (LA-upper_arm)	left_shoulder12	hinge	angle (rad)	
20	Torso-L arm2 angle (LA-upper_arm)	left_shoulder12	hinge	angle (rad)	
21	L upper-lower arm angle	left_elbow	hinge	angle (rad)	
22	Torso x-vel. ctr.	root	free	vel. (m/s)	
23	Torso y-vel. ctr.	root	free	vel. (m/s)	
24	Torso z-vel. ctr.	root	free	vel. (m/s)	
25	Torso x-ang. vel. ctr.	root	free	ang. vel. (rad/s)	
26	Torso y-ang. vel. ctr.	root	free	ang. vel. (rad/s)	
27	Torso z-ang. vel. ctr.	root	free	ang. vel. (rad/s)	
28	Abdomen ang. vel. z lw.	abdomen_z	hinge	ang. vel. (rad/s)	
29	Abdomen ang. vel. y lw.	abdomen_y	hinge	ang. vel. (rad/s)	
30	Abdomen ang. vel. x pelvis	abdomen_x	hinge	ang. vel. (rad/s)	
31	Pelvis-R hip ang. vel. x	right_hip_xyz	hinge	ang. vel. (rad/s)	
32	Pelvis-R hip ang. vel. y	right_hip_z	hinge	ang. vel. (rad/s)	
33	Pelvis-R hip ang. vel. z	right_hip_y	hinge	ang. vel. (rad/s)	
34	R hip-shin ang. vel.	right_knee	hinge	ang. vel. (rad/s)	
35	Pelvis-L hip ang. vel. x	left_hip_xyz	hinge	ang. vel. (rad/s)	
36	Pelvis-L hip ang. vel. y	left_hip_z	hinge	ang. vel. (rad/s)	
37	Pelvis-L hip ang. vel. z	left_hip_y	hinge	ang. vel. (rad/s)	
38	L hip-shin ang. vel.	left_knee	hinge	ang. vel. (rad/s)	
39	Torso ang. vel. coord.1 (RA-shoulder12)	right_shoulder12	hinge	ang. vel. (rad/s)	
40	Torso ang. vel. coord.2 (RA-shoulder12)	right_shoulder12	hinge	ang. vel. (rad/s)	
41	R upper-lower arm ang. vel.	right_elbow	hinge	ang. vel. (rad/s)	
42	Torso ang. vel. coord.1 (LA-shoulder12)	left_shoulder12	hinge	ang. vel. (rad/s)	
43	Torso ang. vel. coord.2 (LA-shoulder12)	left_shoulder12	hinge	ang. vel. (rad/s)	
44	L upper-lower arm ang. vel.	left_elbow	hinge	ang. vel. (rad/s)	

Table 21: The `HumanoidStandup` environment provides a 45-dimensional observation vector capturing detailed information about the agent's torso, limbs, joints, and their respective velocities. The task requires a humanoid agent to rise from a seated or lying position into a standing pose and maintain balance using whole-body coordination. To simulate partial observability, we retain only absolute positional and angular features related to key body parts, while masking out all velocity terms. This configuration imposes significant uncertainty and challenges the agent to infer dynamic state transitions purely from static posture information.

C Reproducibility Details

C.1 Hyperparameters and Training Settings

Environment	Discount γ	GAE λ	Unroll Length	Num Envs
HalfCheetah	0.98	0.92	256	64
Ant	0.98	0.92	128	64
Walker2D	0.995	0.98	128	32
Reacher	0.995	0.98	64	32
Pusher	0.99	0.95	128	32
Humanoid Standup	0.99	0.80	256	32
Hopper	0.99	0.95	64	64
Swimmer	0.98	0.99	64	32

Table 22: Environment-specific rollout and discounting configurations. All values were fixed across algorithms.

Baseline Model Settings. For S5RL and Mamba-1-PPO, all model architecture and optimizer hyperparameters—such as hidden dimensions, number of layers, and SSM size—were adopted directly from the official implementations and original papers without any architectural modification. For GTrXL-PPO, we used the standard gated Transformer architecture as described in the original work, likewise without architectural changes. Only PPO-related training hyperparameters (e.g., learning rate, entropy coefficient, and reward scaling) were tuned per environment via Bayesian optimization. For LSTM-PPO, we used a single-layer LSTM with hidden state size 128 in all experiments. All algorithms, including META and all baselines, were tuned under an identical hyperparameter optimization budget. For each environment, we conducted the same number of sweep trials (30), using the same training horizon, evaluation frequency, and selection metric.

Parameter	Sweep Range / Value
Objective Metric	`eval/episode_reward`
Optimization Method	Bayesian optimization
Learning Rate	log-uniform in $[10^{-4},\ 3 \times 10^{-3}]$
Entropy Coefficient	log-uniform in $[10^{-5},\ 10^{-3}]$
Reward Scaling	Uniform in $[0.1,\ 0.5]$
Clip ϵ	0.2 (fixed)
Num Minibatches	4 (fixed)
Num Update Epochs	4 (fixed)
Boundary pre-activation D	5 (fixed)

Table 23: Bayesian sweep configuration for each environment. All listed hyperparameters are PPO-specific unless otherwise noted; boundary pre-activation D applies only to META/Mamba-2 models. Each sweep comprised 30 trials per environment.

C.2 Computing Resources

Setting	Value
GPU	NVIDIA RTX 4090 (24GB)
Number of GPUs	1
Precision	Full fp32 training
Framework	PyTorch 2.6.0+cu124
RL Libraries	Brax 0.12.3, JAX/JAXlib 0.6.0
OS	Ubuntu 22.04 LTS

Table 24: Computing environment and software dependencies.

C.3 Existing Assets and Licenses

The experiments use open-source benchmark and software assets: Brax (Apache-2.0), POPGym (MIT), Mamba/`mamba-ssm` (Apache-2.0), PyTorch (BSD-style), and JAX/JAXlib (Apache-2.0). The supplemental code documents these licenses and retains attribution for the upstream Mamba files that are patched for META. No proprietary datasets, pretrained models, or restricted-use assets are redistributed.

C.4 Evaluation Protocol

Item	Setting
Evaluation Frequency	Every 5 training steps
Early Stopping	Only used for sweep/pruning
Episode Length	1000 timesteps
Observation Normalization	Yes (running mean & variance)
Reward Normalization	No
Multiple Seeds	[0, 1, 2, 3, 4]
Result Metrics	Final Return, Best Return AUC, Steps to Target
Reporting Format	Mean $\pm$ Standard Deviation

Table 25: Evaluation protocol for reporting results.

C.5 Ablation Experiments

Item	Setting
Variants	META w/o Episodic Forgetting META w/o Memory Rebalancer META w/o MR and EF (ALL)
Same Seeds as Main Exp.	Yes
Same Initialization	Yes
Same Hyperparameters	Yes (only corresponding component ablated)

Table 26: Ablation study settings.