Exploring Lifelong Adaptation: In-Context Reinforcement Learning in Non-Stationary Environments

Ye Wang[1,*] **Kaiqian Cui**[2,*] **Xinrun Xu**[3,*] **Tao Zhang**[4] **Qin Jin**[1,†]
[1]AIM3 Lab, Renmin University of China [2]Sun Yat-sen University
[3]University of the Chinese Academy of Sciences [4]Tsinghua University
[*]Equal contribution. [†]Corresponding author.

Abstract

Standard In-Context Reinforcement Learning (ICRL) typically assumes stationary dynamics within an interaction history, a restriction that limits its real-world applicability. We formalize Lifelong ICRL, a more realistic paradigm in which agents must continuously adapt to diverse non-stationary dynamics, ranging from abrupt random shifts to structured temporal drifts within a single lifetime. To probe the adaptation limits of this setting, we conduct a large-scale empirical study across non-stationary environments that span discrete symbolic reasoning and continuous physics-based control. We systematically investigate three core dimensions governing adaptation: (i) Training Regimes, comparing the generalization boundaries of domain randomization, stationary training, non-stationary training, and their staged combinatorial strategies; (ii) Model Architectures, evaluating modern sequence models, including Linear Attention and hybrid designs; and (iii) Impact of Scale, analyzing the relative contributions of interaction scale and model capacity. Our results reveal how these factors jointly shape robust in-context adaptation under unknown dynamics, yielding actionable insights for the design of generalist agents.

1 Introduction

Deep Reinforcement Learning (RL) has achieved remarkable success in solving individual tasks through trial-and-error interactions [1]. Despite its effectiveness, standard RL typically requires extensive gradient-based training when adapting to each new task. To address this limitation, In-Context Reinforcement Learning (ICRL) has recently emerged as a promising paradigm [2, 3]. By formulating policy improvement as a sequence modeling problem, ICRL enables agents to infer task structures and improve performance purely through forward inference, leveraging interaction histories across multiple episodes within a context window.

However, existing ICRL formulations [2, 4] rely on a restrictive *Episodic Stationarity Assumption*: while tasks may differ across trials, the latent environment dynamics are assumed to remain fixed across all episodes within a single trial. This assumption fundamentally limits the application of ICRL in realistic environments, where dynamics may evolve continuously due to physical parameters drift or change abruptly following interaction rules shift [5]. In such non-stationary settings, historical interaction data can play dual roles: serving as valuable context under structured evolution, or as harmful interference when the underlying dynamics change. An agent relying on standard ICRL lacks an explicit mechanism to distinguish between relevant context and obsolete history, leading to degraded adaptation when latent task structure evolves within a single lifetime.

To overcome this limitation, we formalize **Lifelong In-Context Reinforcement Learning (Lifelong ICRL)**, a principled extension of ICRL where environment dynamics evolve according to a latent

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

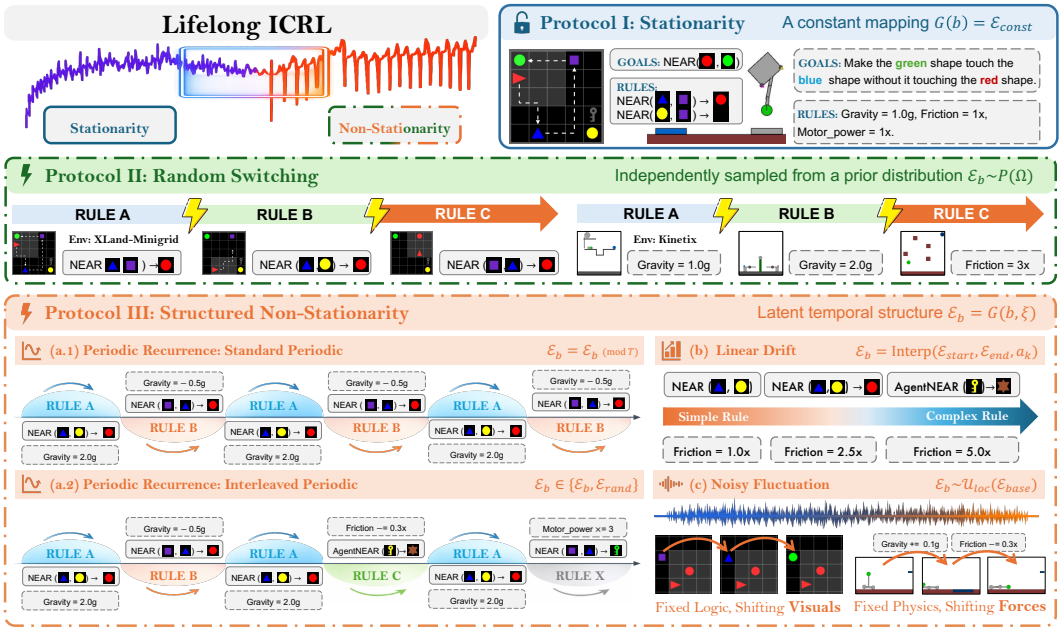


Figure 1: Environments and evaluation protocols for Lifelong ICRL. XLand-Minigrid varies symbolic task rules, while Kinetix varies scene layouts and physical parameters. Across episodes within a lifetime, task configurations remain fixed (Stationary), switch independently (Random Switching), or follow periodic, drifting, or locally perturbed sequences (Structured Non-Stationarity). Task changes neither trigger a memory reset nor provide a change signal, and policy weights remain frozen.

temporal process across episodes. This formulation necessitates a fundamental shift in context utilization: memory retention and forgetting must be treated as **active, history-dependent decisions**, rather than passive accumulation. The Lifelong ICRL agent must not only infer current task parameters, but also reason about the temporal structure governing environmental change.

To systematically study this capability, we introduce a comprehensive evaluation suite built on XLand-Minigrid and Kinetix, spanning discrete symbolic reasoning and continuous control tasks, evaluated under diverse non-stationary protocols. Using this testbed, we conduct an extensive empirical study examining three key dimensions: training regimes (stationary vs. non-stationary curricula), model architectures (Sliding Window Attention, Linear Attention, State Space Models, and hybrid designs), and scaling behavior (interaction steps and model capacity). Our main contributions are summarized as follows:

- We formalize Lifelong ICRL, extending Meta-RL to procedurally evolving Non-Stationary Latent MDPs that require active online inference of environmental dynamics.
- We introduce a novel evaluation suite with controllable intra-lifetime non-stationarity, providing a rigorous and reusable testbed for studying lifelong adaptation.
- We present a systematic empirical analysis of the factors governing Lifelong ICRL, revealing the roles of training regimes, model architectures, and scaling behavior in enabling robust non-stationary adaptation.

2 Related Work

In-Context Reinforcement Learning. ICRL treats meta-reinforcement learning as a sequence modeling problem, where agents improve policies via forward inference without gradient updates at deployment. Pioneered by RL^2 [6] using RNNs, recent approaches leverage Transformers to scale to complex tasks. Algorithm Distillation (AD) [2] distills the learning histories of an RL algorithm, while Decision-Pretrained Transformer (DPT) [3] uses supervised pretraining to predict optimal actions from a query state and an interaction dataset. Subsequent works like AMAGO [4] and AdA [7] extend this to long-horizon memory and open-ended domains. However, these methods

predominantly rely on the *Episodic Stationarity Assumption*, where the latent task context is sampled once and fixed for the entire context window. Our work investigates the **Lifelong ICRL** setting, where the latent context evolves procedurally across episodes, requiring the agent to continuously track dynamics shifts. We train agents through RL using reward feedback without expert action labels. Supervised approaches offer a complementary route, and our training regimes could guide the construction of their pretraining data.

Non-Stationarity and Continual Learning. Addressing evolving environments is a longstanding challenge in RL. Meta-learning approaches support rapid adaptation to changing environments [8, 9]. Online system identification estimates dynamics parameters from recent interactions and supplies them to a universal policy [10]. LILAC [11] uses a latent variable model to represent evolving environments for off-policy RL, while CORRO [12] learns task representations robust to changes in behavior policies in offline meta-RL. *Continual Learning* focuses on preventing catastrophic forgetting across disjoint tasks via regularization [13] or modularity [14], while *Domain Randomization* (DR) [15, 16] promotes robustness by training across randomized environments. We study whether non-stationary adaptation can emerge through context processing within a frozen sequence policy, without a separate model for system identification or latent task evolution.

Sequence Architectures for Decision Making. Transformers [17] enable sequence modeling for decision making, as demonstrated by Decision Transformer [18] and Trajectory Transformer [19] in offline RL. However, the quadratic cost of full self-attention makes long interaction histories expensive to process. Recent linear-time architectures offer efficient alternatives: State Space Models (SSMs) like Mamba [20, 21], linear attention variants like RWKV [22], and fast-weight formulations like DeltaNet [23, 24]. We study how such memory mechanisms support adaptation across episodes when tasks change within a lifetime, requiring both retention of useful experience and revision of outdated context.

3 Problem Formulation

We formalize *Lifelong In-Context Reinforcement Learning* (Lifelong ICRL), extending the standard In-Context RL framework to a procedurally evolving, non-stationary setting that captures long-term environmental changes within a single agent lifetime.

3.1 Background: Standard In-Context RL

Standard In-Context RL (or Meta-RL) [6, 2, 4] is defined over a distribution of tasks. Each task corresponds to a Markov Decision Process parameterized by a latent context variable $z \in \mathcal{Z}$, yielding a Latent MDP (LMDP) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{Z}, \mathcal{T}, \mathcal{R}, \gamma \rangle$, where transitions $\mathcal{T}(s'|s, a; z)$ and rewards $\mathcal{R}(s, a; z)$ depend on z. This formulation can be viewed as a contextual MDP with an unobserved task context.

Because z is not directly observed, the agent operates in a Partially Observable MDP (POMDP) and must infer z from its interaction history $\mathcal{H}_t = (o_0, a_0, r_0, \ldots, o_t)$. In the standard setting, a context $z \sim p(z)$ is sampled at the beginning of a trial and remains fixed across all episodes. The objective is to learn a policy π_θ, typically implemented as a sequence model, that maximizes expected return:

$$\mathcal{J}_{\text{static}}(\theta) = \mathbb{E}_{z\sim p(z)} \left[\mathbb{E}_{\tau\sim\pi_\theta, \mathcal{T}_z} \left[\sum_{t=0}^{H} \gamma^t r(s_t, a_t; z) \right] \right]. \tag{1}$$

Unlike Online RL, which updates parameters θ via gradients, ICRL freezes θ during inference. All adaptation is performed implicitly through the model's forward pass by conditioning on the interaction history. Under the assumption that z is static, this reduces to a form of system identification. However, this assumption fails to capture realistic scenarios in which environmental dynamics evolve over time within a single lifetime.

3.2 Lifelong In-Context Reinforcement Learning

We generalize the standard setting by allowing the latent context to evolve across episodes. Specifically, we define a Lifelong Latent MDP, in which an agent interacts with the environment over a lifetime consisting of K episodes.

Procedural Dynamics Evolution. Instead of sampling a single latent context z, we sample a *dynamics evolution function* ψ from a meta-distribution Ψ at the start of each lifetime. The function $\psi : \mathbb{N} \to \mathcal{Z}$ determines how the environment changes, such that the active context for episode k is $z_k = \psi(k)$, with $\psi(k) \equiv z$ recovering the fixed-task trial described above. Within a lifetime, changes in z_k are not explicitly signaled and do not trigger an external memory reset. The agent must therefore infer these changes and re-adapt from its ongoing interaction history while policy parameters θ remain frozen.

This setting fundamentally changes how context should be used. Lifelong ICRL treats memory retention and forgetting as **active, history-dependent decisions** performed entirely within a frozen model. A Lifelong ICRL agent must infer not only the current latent context z_k, but also the temporal structure governing ψ. When ψ exhibits structure (e.g., smooth drift or periodicity), the agent should leverage long-term context to extrapolate future dynamics; when ψ induces abrupt or random switches, the agent must selectively discard obsolete history. All such adaptation must be achieved through transient inference within a finite context window.

Optimization Objective. The policy $\pi_\theta(a_t|\mathcal{H}_{k,t})$ is conditioned on the lifelong history $\mathcal{H}_{k,t}$, which includes trajectories from episodes $1, \ldots, k-1$ and the partial trajectory of current episode k. The sequence model implicitly acts as a Bayesian filter over both the current state s_t and the latent evolution process $\psi(k)$. The objective is to maximize cumulative return over the lifetime, averaging over all possible evolution functions:

$$\mathcal{J}_{\text{lifelong}}(\theta) = \mathbb{E}_{\psi\sim\Psi}\left[\sum_{k=1}^{K} \mathbb{E}_{\tau_k\sim\pi_\theta(\cdot|\mathcal{H}_k),\mathcal{T}_{\psi(k)}}\left[\sum_{t=0}^{H} \gamma^t r(s_t, a_t; \psi(k))\right]\right]. \tag{2}$$

Optimizing Eq.2 incentivizes the emergence of meta-strategies that exploit temporal structure when present, while remaining robust to unstructured non-stationarity—without explicit supervision over when or how environmental dynamics change.

4 Environments and Evaluation Protocol

To evaluate the adaptation limits of Lifelong In-Context Reinforcement Learning, we construct a benchmark suite covering both discrete symbolic reasoning (XLand-Minigrid) and continuous physics-based control (Kinetix), as illustrated in Figure 1. In both environments, an agent operates within a single prolonged lifetime consisting of K episodes, partitioned into M consecutive blocks. Each block $b \in \{1, \ldots, M\}$ consists of N episodes (such that $K = M \times N$) and is governed by a latent environment configuration $\mathcal{E}_b \in \Omega$. Here, Ω defines the global space of possible task dynamics, corresponding to the latent space $\mathcal{Z}$. We introduce **Intra-lifetime Non-stationarity** by evolving $\mathcal{E}_b$ according to controlled temporal protocols. As the agent's parameters remain frozen, successful performance requires inferring and adapting to dynamic changes solely through in-context history. Additional environment details are provided in Appendix A.

4.1 Symbolic Non-Stationarity: XLand-Minigrid

XLand-Minigrid [25] evaluates symbolic hypothesis testing in grid-world environments, where tasks are defined by latent logical rules governing object interactions and goal conditions. Agents must infer these rules from local symbolic observations via trial-and-error. To introduce Intra-lifetime Non-stationarity, we instantiate the configuration space Ω as a global set of logical rules Ω_{Rules} of varying compositional depths. At the beginning of each block b, the active configuration $\mathcal{E}_b \in \Omega_{\text{Rules}}$ is updated according to the sequence protocol. While the visual grid layout is randomized at every episode to prevent map memorization, the underlying causal logic defined by $\mathcal{E}_b$ remains fixed within block. This design forces the agent to disentangle superficial visual variations from genuine shifts in symbolic dynamics.

4.2 Physics-Based Non-Stationarity: Kinetix

We extend Kinetix [26] to support dynamic changes in physical laws. Each environment configuration is defined as $\mathcal{E} = (\mathcal{L}, \Phi)$, where $\mathcal{L}$ specifies the scene layout (e.g., arrangement of rigid bodies, joints, and thrusters) and Φ denotes physics parameters (e.g., gravity, friction, restitution). Intra-lifetime

non-stationarity is introduced by updating $(\mathcal{L}, \Phi)$ across blocks. Following the original Kinetix design, we further evaluate Out-of-Distribution (OOD) adaptation: training environments are sampled from procedurally generated unstructured morphologies, while evaluation environments consist of human-designed, semantically structured tasks. This setting tests whether dynamics inference learned from chaotic interactions generalizes to meaningful control under drifting physics.

4.3 Sequence Sampling Protocols

To systematically probe adaptation capabilities, we instantiate the dynamics evolution function ψ as a deterministic temporal mapping $G : \mathbb{N} \times \Xi \to \Omega$. Specifically, the configuration sequence $S = \{\mathcal{E}_1, \ldots, \mathcal{E}_M\}$ is generated via $\mathcal{E}_b = G(b; \xi)$, where $\xi \in \Xi$ represents structural hyperparameters. We define three evaluation protocols based on the properties of G, using Stationarity as a fixed-task reference. The non-stationary protocols are motivated by abrupt faults, recurring operating conditions, gradual physical changes such as aging, and local fluctuations.

Protocol I: Stationarity. The configuration remains fixed throughout the lifetime, corresponding to a constant mapping $G(b) \equiv \mathcal{E}_{\text{const}}$. This setting evaluates the asymptotic convergence and long-term stability of the agent in the absence of external perturbations.

Protocol II: Random Switching. This regime is designed to mimic environments characterized by unpredictable volatility. At each block b, the configuration is independently sampled from a prior distribution $\mathcal{E}_b \sim P(\Omega)$. This setting evaluates reactive adaptation, requiring rapid identification of novel dynamics without temporal correlations.

Protocol III: Structured Non-Stationarity. This regime introduces latent temporal structure, probing the agent's capacity for anticipation and long-term adaptation. We consider three representative patterns: (1) **Periodic Recurrence**, which we further categorize into two sub-patterns: (a) *Standard Periodic*, where configurations follow a repeating motif with period T, such that $\mathcal{E}_b = \mathcal{E}_{1+((b-1) \bmod T)}$; and (b) *Interleaved Periodic*, which injects noise slots of length Δ between motifs to stress-test memory robustness. Let $L = T + \Delta$ be the total cycle length, this is formalized as:

$$\mathcal{E}_b = \begin{cases} \mathcal{E}_{1+((b-1) \bmod L)} & \text{if } ((b-1) \bmod L) < T \\ \mathcal{E}_{\text{rand}} & \text{otherwise} \end{cases} \tag{3}$$

where $\mathcal{E}_{\text{rand}}$ denotes a randomly sampled configuration; (2) **Linear Drift**, where configurations evolve smoothly along a trajectory in parameter space, formalized as $\mathcal{E}_b = \text{Interp}(\mathcal{E}_{\text{start}}, \mathcal{E}_{\text{end}}, \alpha_b)$, with α_b denoting the interpolation factor; and (3) **Noisy Fluctuation**, where the underlying task semantics remain anchored to a base configuration $\mathcal{E}_{\text{base}}$ but are subject to stochastic local perturbations. This is formalized as $\mathcal{E}_b \sim \mathcal{U}_{\text{loc}}(\mathcal{E}_{\text{base}})$, where $\mathcal{U}_{\text{loc}}$ represents a domain-specific uncertainty set. This encompasses both continuous parametric jitter in physics-based control and superficial visual or layout variances in symbolic reasoning.

4.4 Evaluation Metrics

Standard average returns fail to reveal the temporal dynamics of lifelong adaptation. To address this limitation, we introduce a decoupled metric suite that explicitly disentangles baseline competence, contextual benefit, and adaptation dynamics. Let $R_{b,i}$ denote the return obtained in the i-th episode of block b, where each block consists of N episodes governed by a latent configuration $\mathcal{E}_b$.

Foundational Performance. We first define the ***Zero-Shot Return*** (R_{ZS}) as the agent's baseline performance in the absence of valid historical context. Formally, for any configuration $\mathcal{E}$, $R_{\text{ZS}}(\mathcal{E})$ denotes the expected return of the initial episode when $\mathcal{E}$ is encountered without any prior task-specific interaction. Within each evaluation protocol, R_{ZS} is averaged over the configurations encountered in that protocol. The ***In-Sequence Return*** (R_{Seq}) captures overall lifetime performance by averaging returns across all episodes: $R_{\text{Seq}} = \mathbb{E}_{b,i}[R_{b,i}]$. To isolate the net benefit induced by contextual information, we define the ***Contextual Gain*** (Δ_{Ctx}) as the average improvement over the zero-shot baseline of the corresponding configuration:

$$\Delta_{\text{Ctx}} = \frac{1}{MN} \sum_{b=1}^{M} \sum_{i=1}^{N} (R_{b,i} - R_{\text{ZS}}(\mathcal{E}_b)) \tag{4}$$

For both XLand-Minigrid and Kinetix, these metrics are computed within each sub-protocol before averaging over the sub-protocols in a reported protocol group. In the training-regime and architecture

comparisons (Tables 1 and 4), each gain therefore uses the zero-shot baselines of its corresponding evaluation configurations.

Adaptation Dynamics. Beyond average performance, we explicitly characterize adaptation behavior at regime transitions and during stationary phases. We define two differential metrics, both measured relative to the zero-shot baseline. ***Switch Resilience*** ($\Delta_{\textbf{\textit{Switch}}}$) quantifies robustness to dynamic shifts by measuring the performance gain in the *first* episode following a configuration switch. This metric isolates the agent's ability to mitigate interference from outdated context:

$$\Delta_{\text{Switch}} = \frac{1}{M-1} \sum_{b=2}^{M} (R_{b,1} - R_{\text{ZS}}(\mathcal{E}_b)) \tag{5}$$

Conversely, ***Stationary Adaptation*** ($\Delta_{\textbf{\textit{Static}}}$) captures learning efficiency within stable regimes. It averages the performance gain over all *non-switch* episodes (i.e., episodes $k \geq 2$ within each block), reflecting how effectively the agent exploits in-context updates once the new configuration is established:

$$\Delta_{\text{Static}} = \frac{1}{M(N-1)} \sum_{b=1}^{M} \sum_{i=2}^{N} (R_{b,i} - R_{\text{ZS}}(\mathcal{E}_b)) \tag{6}$$

Positive values of Δ_{Switch} and Δ_{Static} indicate effective in-context adaptation in the corresponding phases. When no task switch occurs ($M = 1$), or no stable-phase episode exists ($N = 1$), the corresponding phase gain is reported as zero by convention.

5 Experiments

We evaluate Lifelong ICRL (LICRL) performance in symbolic and physics-based environments, focusing on how training meta-distributions, architectures, and interaction scales determine adaptation. Throughout this section, a training regime specifies how tasks are sampled and evolve during training. An evaluation protocol specifies how the latent context z_k evolves during a test lifetime with policy parameters θ frozen.

5.1 Experimental Setup

Environments and Task Complexity. We conduct experiments on two challenging benchmarks: (1) **XLand-Minigrid**: A symbolic grid-world where task rules (e.g., "put green object near yellow door") are non-stationary. We train on the Medium task distribution with rule recursion depth at most 2. Under Stationary, Random Switching, Periodic Recurrence, and Noisy Fluctuation, we evaluate on held-out Out-of-Distribution (OOD) *depth-3* rules, which require higher-order logical reasoning and composition. Linear Drift evaluates task sequences with progressively increasing rule depth and numbers of distractors. (2) **Kinetix**: A 2D physics-based environment where agents interact with joints, motors, and rigid bodies. We train on procedurally generated unstructured layouts with randomized physics parameters and evaluate on held-out, human-designed *structured layouts* that test the agent's ability to generalize motor control to unseen mechanical configurations and randomized physical constants.

Lifelong Evaluation Protocol. An evaluation lifetime consists of 200 episodes in XLand-Minigrid or 100 episodes in Kinetix. The default evaluation uses 4096 parallel environments for XLand-Minigrid and 2048 for Kinetix, with returns averaged across environments. Within a lifetime, the latent context z_k follows four evaluation protocols: (1) **Stationary**: z_k remains fixed throughout the lifetime. (2) **Random Switch**: z_k is independently resampled every 1 or 10 episodes. (3) **Periodic Recurrence**: z_k cycles through a recurring sequence, testing memory and task recognition. (4) **Drift**: Linear Drift and Noisy Fluctuation test adaptation to gradual changes and local perturbations, respectively.

Agent and Training Details. The default backbone is **Gated DeltaNet** ($d_{\text{model}} = 512, L = 6, \approx 20.4$M parameters). The default training budgets are 10B environment steps for XLand-Minigrid and 1B for Kinetix. We train using PPO and the Adam optimizer, with default initial learning rates of 10^{-3} for XLand-Minigrid and 3×10^{-4} for Kinetix. The discount factor is $\gamma = 0.99$ in both environments. Additional training hyperparameters are provided in Table 10.

5.2 Dimension 1: Training Regimes and Meta-Learning

We use Gated DeltaNet, a representative recurrent model based on linear attention, to compare training regimes with the architecture held fixed. We evaluate six regimes that differ in how tasks evolve during training (Table 1): (1) **Domain Randomization (DR)**: Dynamics z_k are resampled every episode, effectively penalizing reliance on cross-episodic history. (2) **Stationary**: z_k is fixed for the entire 10^5-step training lifetime, following the fixed-task trial structure of standard meta-RL. (3) **Non-Stationary (NS)**: z_k switches to a random configuration at unpredictable intervals, simulating an unstable world. (4) **Composite Non-Stationary (CNS)**: A sophisticated mixture of random switches and *periodic recurrences*, designed to force the agent to distinguish between novel and reappearing tasks. (5) **Curriculum (DR $\rightarrow$ Stat)**: Pre-training under high-noise (DR) followed by stationary fine-tuning. (6) **Curriculum (DR $\rightarrow$ NS)**: DR pre-training followed by non-stationary fine-tuning to test if context-awareness can be retroactively learned.

Table 1: Performance of training paradigms across evaluation protocols. We report the local zero-shot baseline (R_{ZS}), sequence return (R_{Seq}), and adaptation metrics (Δ). Best results bolded.

Training Regime	**Base**	**I. Stationary**		**II. Random Switch**			**III. Structured Non-Stationarity**			
							Periodic Recurrence		*Drift (Linear+Noisy)*	
	R_{ZS}	R_{Seq}	Δ_{Static}	R_{Seq}	Δ_{Switch}	Δ_{Static}	R_{Seq}	Δ_{Ctx}	R_{Seq}	Δ_{Ctx}
XLand-Minigrid (Symbolic)										
(1) Domain Randomization	0.141	0.128	-0.021	0.067	-0.103	-0.020	0.058	-0.046	0.159	-0.009
(2) Stationary	0.149	0.154	-0.003	0.048	-0.112	-0.038	0.061	-0.050	0.201	0.020
(3) Non-Stationary	0.142	0.165	0.014	0.145	-0.018	0.006	0.104	-0.001	0.199	0.028
(4) Composite Non-Stationary	**0.150**	**0.181**	**0.017**	**0.153**	-0.023	**0.008**	**0.117**	**0.003**	**0.214**	**0.042**
(5) Curr: DR $\rightarrow$ Stat	0.119	0.119	-0.007	0.030	-0.098	-0.037	0.036	-0.045	0.166	0.014
(6) Curr: DR $\rightarrow$ Non-Stat	0.132	0.150	0.009	0.132	**-0.017**	0.005	0.096	0.001	0.183	0.021
Kinetix (Physics-Based)										
(1) Domain Randomization	0.005	0.003	-0.002	-0.018	0.000	-0.001	-0.019	0.001	-0.006	-0.002
(2) Stationary	0.018	0.018	0.000	-0.002	-0.003	-0.001	0.002	-0.002	0.019	0.000
(3) Non-Stationary	**0.037**	0.020	-0.018	**0.000**	-0.019	-0.013	0.002	-0.008	0.013	0.003
(4) Composite Non-Stationary	0.011	**0.022**	**0.011**	-0.001	**0.004**	**0.008**	**0.004**	**0.011**	**0.028**	**0.008**
(5) Curr: DR $\rightarrow$ Stat	0.012	0.003	-0.009	-0.014	0.000	0.000	-0.007	0.003	-0.007	-0.006
(6) Curr: DR $\rightarrow$ Non-Stat	0.010	0.005	-0.005	-0.014	0.002	0.001	-0.005	0.002	-0.003	-0.001

Stationary Training and Adaptation to Switches. The Stationary-trained agent achieves a competitive zero-shot baseline in XLand-Minigrid ($R_{\text{ZS}} = 0.149$), comparable to CNS (0.150) in Table 1. However, under Random Switch, it achieves only $R_{\text{Seq}} = 0.048$, with $\Delta_{\text{Switch}} = -0.112$. The negative Δ_{Switch} indicates that retained history reduces performance relative to the zero-shot baseline immediately after a task change. The separately trained Stationary agent in Kinetix also obtains a low return under Random Switch ($R_{\text{Seq}} = -0.002$). These observations suggest that learning to exploit context for a fixed task does not automatically confer the ability to update that context when the task changes.

Composite Training and Generalization. Introducing task changes during training improves Random Switch performance in XLand, with NS and CNS reaching $R_{\text{Seq}} = 0.145$ and 0.153, respectively, compared with 0.048 for Stationary training (Table 1). Within ten-episode Random Switch blocks in XLand, CNS achieves higher success rates and reaches goals in fewer steps on average over episodes 1–9 than in the first episode after a switch (Table 2). CNS also performs well under stationary evaluation ($R_{\text{Seq}} = 0.181$, $\Delta_{\text{Static}} = 0.017$) and on the unseen Drift protocol ($R_{\text{Seq}} = 0.214$, $\Delta_{\text{Ctx}} = 0.042$). For the separately trained Kinetix agents, CNS achieves a higher Drift return than NS (0.028 vs. 0.013), with both regimes yielding positive context gains. We hypothesize that exposure to periodic recurrence helps agents exploit temporal structure beyond the training protocols. To assess robustness across training runs, we repeat the XLand comparison of Stationary, NS, and CNS over five seeds and report the results and statistical analysis in Appendix C (Table 11). These results support the advantage of non-stationary training on the switching protocols, with CNS achieving the highest mean R_{Seq} among the three regimes under all four protocols.

Table 2: CNS behavior in ten-episode XLand Random Switch blocks.

Metric	**Episode 0**	**Episodes 1–9**
Success rate (%)	24.8	28.0
Steps to goal	128.8	104.7
Forward-action share (%)	34.9	36.4

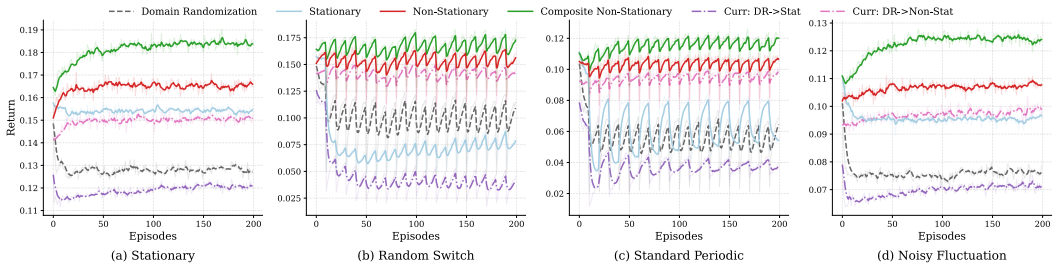


Figure 2: Per-episode returns of different training paradigms. CNS demonstrates the most robust adaptation across both symbolic and physics-based OOD task sequences.

Table 4: Performance of sequence architectures under the *Non-Stationary* regime. Best results bolded.

Architecture	**Base**	**I. Stationary**		**II. Random Switch**			**III. Structured Non-Stationarity**			
							Periodic Recurrence		*Drift (Linear+Noisy)*	
	R_{ZS}	R_{Seq}	Δ_{Static}	R_{Seq}	Δ_{Switch}	Δ_{Static}	R_{Seq}	Δ_{Ctx}	R_{Seq}	Δ_{Ctx}
XLand-Minigrid (Symbolic)										
(1) Gated DeltaNet	0.142	0.165	0.014	0.145	-0.018	0.006	0.104	-0.001	0.199	0.028
(2) Sliding Window Attn	0.057	0.073	**0.021**	0.048	**-0.017**	**0.008**	0.035	**0.007**	0.103	0.024
(3) Hybrid	**0.154**	0.171	0.005	**0.153**	-0.024	0.003	0.110	-0.003	0.203	0.018
(4) Mamba2	0.150	**0.175**	0.018	0.147	-0.025	0.003	**0.112**	0.001	**0.210**	**0.032**
Kinetix (Physics-Based)										
(1) Gated DeltaNet	0.037	0.020	-0.018	0.000	-0.019	-0.013	0.002	-0.008	0.013	0.003
(2) Sliding Window Attn	0.046	0.052	0.008	0.046	-0.010	0.001	0.031	-0.015	0.041	-0.005
(3) Hybrid	0.071	0.062	0.002	0.083	**0.005**	0.005	0.066	0.001	0.084	0.004
(4) Mamba2	**0.074**	**0.084**	**0.023**	**0.088**	0.004	**0.009**	**0.076**	**0.005**	**0.092**	**0.011**

Persistent Effects of DR Pretraining. The DR $\rightarrow$ NS curriculum tests whether introducing non-stationarity only after DR pretraining is sufficient to learn effective context use. In XLand Periodic Recurrence, DR $\rightarrow$ NS reaches $R_{Seq} = 0.096$, below NS (0.104) and CNS (0.117) trained without DR pretraining (Table 1). For the separately trained Kinetix agents, the same curriculum reaches $R_{Seq} = 0.005$ under stationary evaluation, compared with 0.020 for NS training without DR pretraining. To understand this gap, we inspect the gate that controls how much information the XLand agents retain across episodes, focusing on Gated DeltaNet's bottom layer. A near-zero gate value means that this layer's memory is almost cleared when a new episode begins. This occurs at 49% of episode boundaries for DR and 58% for DR $\rightarrow$ NS, compared with 20% for NS. The higher frequency of clearing this memory after DR $\rightarrow$ NS training is consistent with a forgetting bias that persists within the tested budget, highlighting the importance of early exposure to non-stationarity.

5.3 Analysis of Hidden State Dynamics

To understand how the training regimes shape adaptation, we examine their layer-wise hidden-state responses to task changes. For each layer, Δ is the mean hidden-state cosine similarity during stable periods minus that at task switches. Positive values indicate more pronounced state changes at switches.

Table 3: State similarity shift (Δ).

Layer	**Stat**	**NS**	**CNS**
L0 (Bot)	+0.004	+0.013	+0.010
L2 (Mid)	-0.002	+0.087	+0.069
L3 (Mid)	-0.001	+0.081	+0.074
L5 (Top)	-0.000	+0.116	+0.065

In Table 3, Stationary training yields $\Delta \approx 0$ across the four reported layers, while NS and CNS show larger positive shifts in the middle and top layers. This distinction is reproduced in the five-seed results in Appendix C.1.2 (Table 20), where Stationary remains near zero at L5 and NS and CNS show positive shifts at L3 and L5. These observations suggest that non-stationary training encourages agents to revise their internal context when tasks change, consistent with the adaptation gains observed above.

5.4 Dimension 2: Model Architectures and Recurrence

To isolate architectural biases, we fix the training distribution to the **NS** regime and evaluate four representative sequence backbones: (1) **Gated DeltaNet**: A linear-attention-based recurrent model

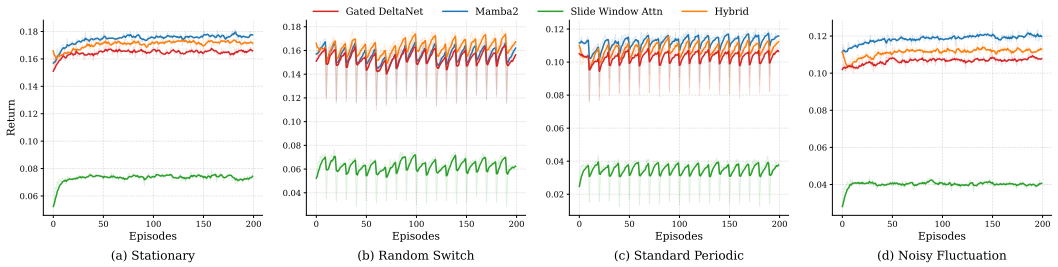


Figure 3: Per-episode returns of sequence backbones. Recurrent models bridge dependencies across 200 episodes, while windowed attention suffers from limited memory.

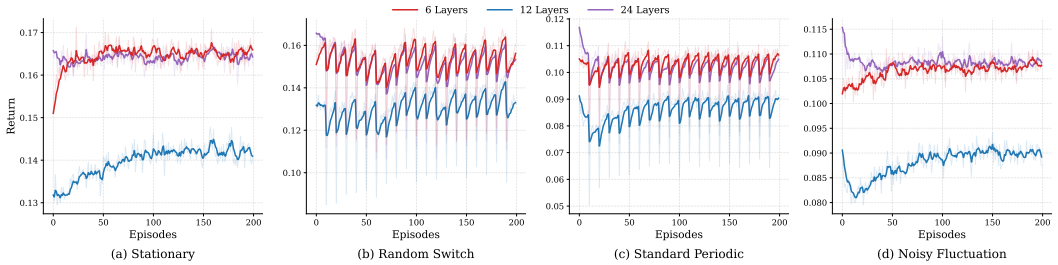


Figure 4: Per-episode returns for varying network depths (6, 12, 24 layers).

that uses an outer-product update rule to compress history into a fast-weight matrix. (2) **Mamba2**: A State Space Model (SSM) that utilizes structured semi-separable matrices to model latent dynamics with highly efficient recurrent state updates. (3) **Sliding Window Attention (SWA)**: A Transformer variant with a fixed receptive field of 512 tokens, representing non-recurrent, bounded-memory models. (4) **Hybrid**: A progressive architecture that alternates between Gated DeltaNet layers (for long-term task tracking) and SWA layers (for precise local observation retrieval).

Limitations of Sliding Window Attention. On XLand-Minigrid, SWA yields lower sequence returns than the recurrent and hybrid backbones across all four evaluation protocols (Table 4). Under Random Switch, for example, its R_{Seq} is 0.048, compared with 0.145 for Gated DeltaNet. The gap is already evident in zero-shot performance ($R_{\text{ZS}} = 0.057$ for SWA versus 0.142 for Gated DeltaNet), suggesting that SWA has greater difficulty fitting the meta-training distribution. In Kinetix, the separately trained SWA agent also trails Hybrid and Mamba2 under Random Switch, with $R_{\text{Seq}} = 0.046$, compared with 0.083 for Hybrid and 0.088 for Mamba2.

Hybrid Architecture. Combining Gated DeltaNet with local attention, Hybrid achieves higher sequence returns than either Gated DeltaNet or SWA alone across all four protocols in both environments (Table 4). In XLand, it achieves the highest zero-shot return ($R_{\text{ZS}} = 0.154$) and Random Switch sequence return ($R_{\text{Seq}} = 0.153$) among the four backbones. For the separately trained Kinetix agents, Hybrid and Mamba2 both achieve positive Δ_{Switch} ($+0.005$ and $+0.004$, respectively), with Mamba2 yielding the highest sequence return under each protocol. Figure 3 shows the per-episode returns of the four backbones. To assess robustness across training runs, we repeat the XLand comparison of Gated DeltaNet, SWA, and Hybrid over five training seeds (Appendix C.2, Table 21). Hybrid achieves the highest mean R_{Seq} among these three architectures under all four protocols, supporting the benefit of combining recurrent state with local attention.

Table 5: Training budget comparison (24 layers, XLand, Random Switch).

Training steps	R_{Seq}
10B	0.146
20B	0.155

5.5 Dimension 3: Impact of Scale

Depth Scaling. At a fixed interaction budget, we compare models with 6, 12, and 24 layers (Figure 4). Increasing depth alone does not yield monotonic performance gains. To examine whether more training interaction benefits the deeper model, we supplement this analysis with a 20B-step training

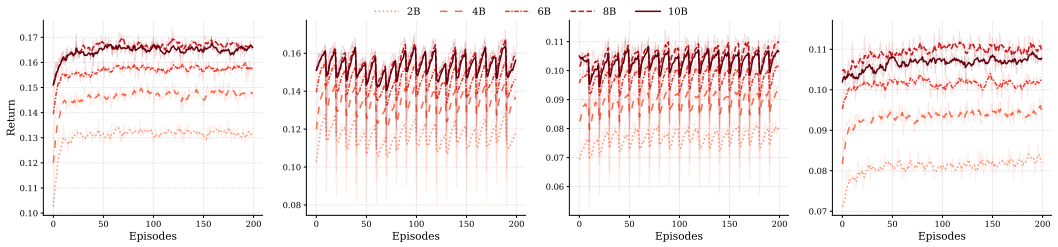


Figure 5: Per-episode returns for different interaction scales (2B to 10B training steps).

budget for the 24-layer model. Under Random Switch in XLand, the 24-layer model obtains a higher sequence return with the larger interaction budget (Table 5).

Interaction Volume. Increasing the training interaction budget from 2B to 10B steps generally improves per-episode returns (Figure 5). Together, the depth and interaction experiments highlight the importance of matching model capacity with sufficient training interaction for lifelong adaptation.

5.6 Comparison with Gradient-Based Adaptation

We compare Frozen LICRL with Online PPO and PPO+EWC under Random Switch in XLand-Minigrid (Table 6). The reported adaptation gains are means $\pm$ standard deviations over 32 independent single-environment runs per method. Frozen LICRL achieves a less negative Δ_{Switch} and a positive Δ_{Static} ($+0.011$), with much lower variability across runs ($\sigma = 0.001$) than the gradient-based baselines ($\sigma \approx 0.24$). These results indicate that frozen-weight in-context adaptation handles task changes more effectively and with greater stability than online gradient updates.

Table 6: Comparison with gradient-based baselines under Random Switch.

Method	R_{ZS}	Δ_{Switch}	Δ_{Static}
Online PPO	0.172	-0.079 ± 0.237	-0.085 ± 0.244
Online PPO + EWC	**0.181**	-0.100 ± 0.250	-0.095 ± 0.253
Frozen LICRL (Ours)	0.164	$\mathbf{-0.026 \pm 0.001}$	$\mathbf{+0.011 \pm 0.001}$

6 Conclusion

We formalize Lifelong In-Context Reinforcement Learning and develop an evaluation suite spanning symbolic reasoning and continuous control under intra-lifetime non-stationarity. Our findings show that effective lifelong adaptation depends on learning both to retain useful experience and to revise outdated context. Non-stationary training helps develop this ability, while composite training supports generalization to unseen patterns of change. Recurrent and hybrid architectures provide effective mechanisms for maintaining and updating context, and the scaling results highlight the importance of matching model capacity with sufficient interaction. Together, these findings offer practical guidance for developing agents that adapt continually through context without deployment-time fine-tuning.

Acknowledgments and Disclosure of Funding

We thank the anonymous reviewers for their valuable comments and suggestions.

Funding. This work was partially supported by the National Natural Science Foundation of China (Grant No. 62576347).

Competing interests. The authors declare no competing interests.

References

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

[2] Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, et al. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*, 2022.

[3] Jonathan Lee, Annie Xie, Aldo Pacchiano, Yash Chandak, Chelsea Finn, Ofir Nachum, and Emma Brunskill. Supervised pretraining can learn in-context reinforcement learning. *Advances in Neural Information Processing Systems*, 36:43057–43083, 2023.

[4] Jake Grigsby, Linxi Fan, and Yuke Zhu. Amago: Scalable in-context reinforcement learning for adaptive agents. *arXiv preprint arXiv:2310.09971*, 2023.

[5] Pouya Hamadanian, Arash Nasr-Esfahany, Malte Schwarzkopf, Siddartha Sen, and Mohammad Alizadeh. Online reinforcement learning in non-stationary context-driven environments. *arXiv preprint arXiv:2302.02182*, 2023.

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

[7] Adaptive Agent Team, Jakob Bauer, Kate Baumli, Satinder Baveja, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, et al. Human-timescale adaptation in an open-ended task space. *arXiv preprint arXiv:2301.07608*, 2023.

[8] Maruan Al-Shedivat, Trapit Bansal, Yura Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[9] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[10] Wenhao Yu, Jie Tan, C. Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. In Nancy M. Amato, Siddhartha S. Srinivasa, Nora Ayanian, and Scott Kuindersma, editors, *Robotics: Science and Systems XIII, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, July 12-16, 2017*, 2017.

[11] Annie Xie, James Harrison, and Chelsea Finn. Deep reinforcement learning amidst lifelong non-stationarity. *arXiv preprint arXiv:2006.10701*, 2020.

[12] Haoqi Yuan and Zongqing Lu. Robust task representations for offline meta-reinforcement learning via contrastive learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 25747–25759. PMLR, 2022.

[13] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[14] Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research*, 75:1401–1476, 2022.

[15] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[16] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, pages 1–8. IEEE, 2018.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[18] 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. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[19] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

[20] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[21] Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

[22] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, Jiaju Lin, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Johan S. Wind, Stanislaw Wozniak, Zhenyuan Zhang, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. RWKV: reinventing rnns for the transformer era. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, volume EMNLP 2023 of *Findings of ACL*, pages 14048–14077. Association for Computational Linguistics, 2023.

[23] Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *arXiv preprint arXiv:2406.06484*, 2024.

[24] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. *arXiv preprint arXiv:2412.06464*, 2024.

[25] Alexander Nikulin, Vladislav Kurenkov, Ilya Zisman, Artem Agarkov, Viacheslav Sinii, and Sergey Kolesnikov. Xland-minigrid: Scalable meta-reinforcement learning environments in jax. *Advances in Neural Information Processing Systems*, 37:43809–43835, 2024.

[26] Michael Matthews, Michael Beukman, Chris Lu, and Jakob Foerster. Kinetix: Investigating the training of general agents through open-ended physics-based control tasks. *arXiv preprint arXiv:2410.23208*, 2024.

Appendix

This appendix provides additional details and extended analyses supporting the main paper. We first present comprehensive descriptions of the evaluation environments and protocols in both the discrete symbolic domain and the continuous physics domain (§A), including XLand-Minigrid specifications (§A.1), Kinetix configurations (§A.2), and the implementation details of the sequence sampling protocols. We then describe the full experimental setup (§B), covering evaluation settings (§B.1), input representations and encoders (§B.2), sequence modeling backbones (§B.3), detailed training hyperparameters (§B.4), and compute resources (§B.5). Finally, we provide extensive additional experimental results and analyses (§C), including fine-grained quantitative evaluations across different training regimes, switching patterns, and structured non-stationary settings, together with detailed tables and visualizations that further substantiate the findings in the main paper. We conclude with a discussion of limitations and broader impacts (§D).

A Environment and Evaluation Protocol Details

To rigorously evaluate the adaptation boundaries of Lifelong In-Context Reinforcement Learning (ICRL), we instantiate our evaluation suite on two distinct domains: XLand-Minigrid, representing discrete symbolic reasoning, and Kinetix, representing continuous physics-based control.

A.1 Discrete Symbolic Domain: XLand-Minigrid

We utilize XLand-Minigrid [25] to construct a partial-observable environment where the underlying causal logic is a latent variable. The environment consists of an $N \times N$ grid (9×9) populated with objects and an agent. The observation space is a local $5 \times 5 \times 2$ tensor centered on the agent, encoding categorical integers for **Object Types** and **Colors**. Crucially, this symbolic observation provides no explicit information regarding the current task logic; the agent must infer the active rules $\mathcal{R}$ and goal G solely through interaction history. The action space consists of seven discrete actions: navigation (move forward, turn left/right), object manipulation (pick up, put down, toggle), and a termination signal. Figure 6 visualizes this inference process.

Atomic Elements and Task Logic. The complexity of the environment stems from the combinatorial explosion of rules and goals. We employ the full set of atomic elements provided by the engine. The tile types and colors used in the observation encoding are listed in Table 7. The task logic is defined by a dependency tree generated via backward chaining. The root of the tree is a specific **Goal** condition (sampled from Table 8), which requires satisfying a logical predicate such as the agent holding a specific object or two objects being adjacent. To satisfy the goal, the agent may need to trigger a sequence of **Production Rules** (sampled from Table 8). For instance, a rule might dictate that placing a *Red Ball* near a *Blue Square* transforms them into a *Green Key*.

Procedural Generation and Difficulty Splits. Training and evaluation use separate task sets. For training, we utilize the Medium-1m task distribution. Tasks in this set are generated with a maximum recursion depth of 2, involving straightforward object dependencies and a limited number of distractor rules (valid rules that lead to dead ends). Evaluation combines held-out depth-3 rules from High-1m with procedurally generated sequences for structured non-stationarity. These sequences retain depth-3 rule structure under Periodic Recurrence and Noisy Fluctuation, while Linear Drift progressively increases rule depth and the number of distractors across blocks.

A.2 Continuous Physics Domain: Kinetix

Kinetix [26] serves as our continuous control testbed, simulating 2D rigid-body dynamics with realistic collision and joint constraints. An environment configuration $\mathcal{E} = (\mathcal{L}, \Phi)$ is defined by the morphological layout $\mathcal{L}$ and the physics parameters Φ. Agents observe the world through $125 \times 125 \times 3$ RGB images and exert control via a continuous action vector $\mathcal{A} \in [-1, 1]^n$, mapping to motor torques and thruster forces.

Procedural Morphology Generation. To prevent the agent from overfitting to specific body structures (e.g., a "walker"), we implement a rigorous procedural generation pipeline for the training distribution. The generation of a random layout $\mathcal{L}$ follows a constructive graph growth algorithm: (1) A root polygon or circle is instantiated at the center; (2) The algorithm iteratively selects an existing

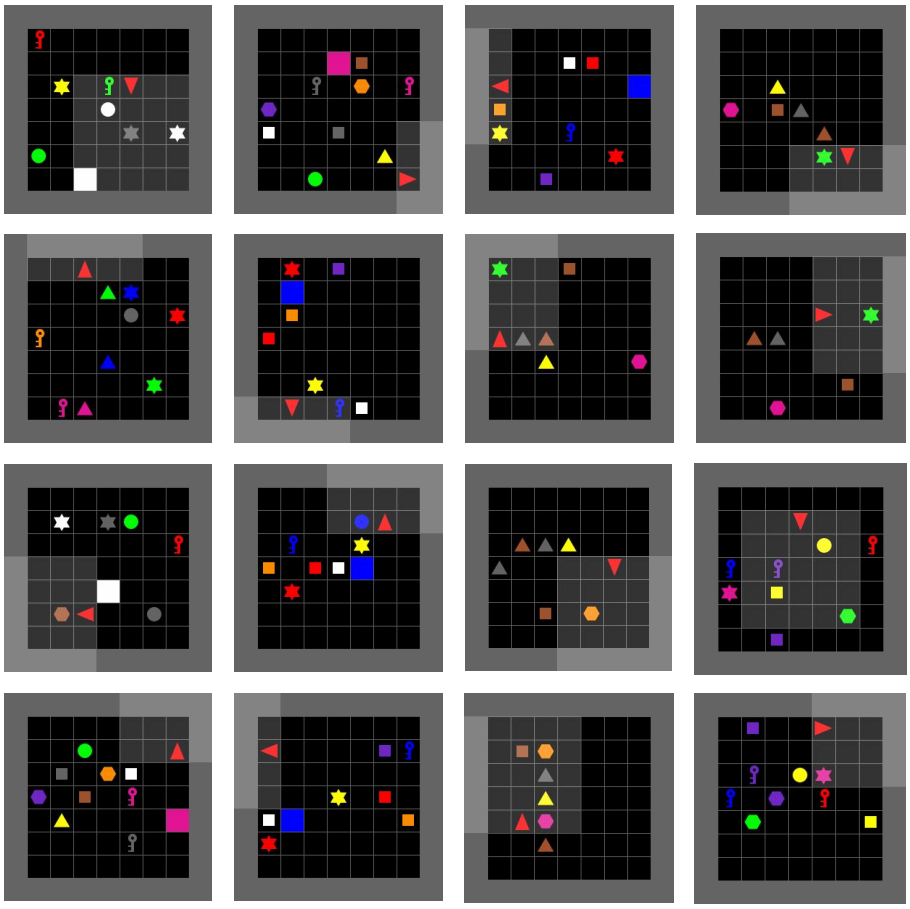


Figure 6: **XLand-Minigrid Environment.** The agent (red triangle) navigates a grid to manipulate objects. While the visual layout (walls, object positions) is randomized every episode, the underlying rules governing object interactions change only periodically, creating blocks of stationarity within a non-stationary lifetime.

body and an attachment point, sampling a new rigid body to connect via a revolute or fixed joint; (3) Actuators (motors and thrusters) are probabilistically attached to joints and bodies; (4) A rejection sampling step verifies the validity of the generated morphology, discarding configurations with initial self-collisions or unstable kinematic chains. This results in a diverse distribution of **Unstructured Morphologies** (Figure 7) that require the agent to learn generalized locomotion policies. Conversely, for evaluation, we utilize **Structured Morphologies** designed by humans (e.g., Bipedal Walker, Hopper, see Figure 8), assessing the transfer of physical reasoning to semantically structured tasks.

Physics Parameter Space. The dynamics of the environment are governed by the parameter vector Φ, which includes both global and local properties. As detailed in Table 9, global parameters control environmental constants such as gravity and air resistance, while local parameters define material properties like friction, restitution (bounciness), and density for individual objects. In our non-stationary protocols, these parameters are latent variables that evolve over time, requiring the agent to continuously perform system identification.

A.3 Implementation of Sequence Sampling Protocols

We now formally map the specific implementation details of both domains to the three evaluation protocols defined in Section 4.

Table 7: **XLand-Minigrid Atomic Elements.** The observation consists of integer IDs mapping to these types and colors.

Tile Type	ID	Color	ID	Tile Type	ID	Color	ID
End of Map	0	End of Map	0	Goal	8	Grey	8
Unseen	1	Unseen	1	Key	9	Black	9
Empty	2	Empty	2	Door (Locked)	10	Orange	10
Floor	3	Red	3	Door (Closed)	11	White	11
Wall	4	Green	4	Door (Open)	12	Brown	12
Ball	5	Blue	5	Hex	13	Pink	13
Square	6	Purple	6	Star	14		
Pyramid	7	Yellow	7				

Table 8: **Supported Goals and Production Rules.** Variables a, b represent objects, c represents the resulting object.

Goal Predicate	ID	Production Rule Logic	ID
`EmptyGoal` (False)	0	`EmptyRule` (No Op)	0
`AgentHoldGoal(a)`	1	`AgentHoldRule(a)` $\rightarrow$ `c`	1
`AgentOnTileGoal(a)`	2	`AgentNearRule(a)` $\rightarrow$ `c`	2
`AgentNearGoal(a)`	3	`TileNearRule(a, b)` $\rightarrow$ `c`	3
`TileNearGoal(a, b)`	4	`TileNearUpRule(a, b)` $\rightarrow$ `c`	4
`AgentOnPositionGoal(x,y)`	5	`TileNearRightRule(a, b)` $\rightarrow$ `c`	5
`TileOnPositionGoal(a,x,y)`	6	`TileNearDownRule(a, b)` $\rightarrow$ `c`	6
`TileNear{Dir}Goal(a, b)`	7-10	`TileNearLeftRule(a, b)` $\rightarrow$ `c`	7
`AgentNear{Dir}Goal(a)`	11-14	`AgentNear{Dir}Rule(a)` $\rightarrow$ `c`	8-11

Protocol I: Stationary Assessment. This protocol establishes a baseline for asymptotic performance. In **XLand-Minigrid**, the underlying rule set $\mathcal{R}$ and goal definition G remain constant throughout the lifetime L, although the grid layout (wall/object positions) is randomized per episode. In **Kinetix**, both the morphology layout $\mathcal{L}$ (e.g., a standard Walker) and the physics parameters Φ are fixed constant.

Protocol II: Random Switching. This regime tests reactive adaptation to unpredictable changes via Independent and Identically Distributed (I.I.D.) sampling. In **XLand-Minigrid**, at the start of each block b, a new rule set $\mathcal{R}_b$ is sampled uniformly from the training or evaluation distribution. We evaluate two variants: *Fast Switching* ($K_b = 1$), and *Slow Switching* ($K_b = 10$). In **Kinetix**, each block uses a configuration $(\mathcal{L}_b, \Phi_b)$ sampled from the held-out structured task pool. Both morphology and physics can therefore change across blocks, requiring the agent to adapt to a new task configuration.

Protocol III: Structured Non-Stationarity. This protocol evaluates the agent's ability to model latent temporal structures, instantiated via three distinct patterns.

(1) Periodic Recurrence. The configuration sequence follows a repeating pattern $\mathcal{E}_b = \mathcal{E}_{1+((b-1) \bmod T)}$. We evaluate two distinct sub-patterns across both domains: **(1) Standard Periodic:** The environment cycles through a fixed set of discrete configurations (e.g., A $\rightarrow$ B $\rightarrow$ A). In **XLand-Minigrid**, this corresponds to alternating between specific rule sets. In **Kinetix**, the environment cycles through specific pairs of morphologies and physics parameters $(\mathcal{L}, \Phi)$ (e.g., alternating between a Standard Walker with High Gravity and a Hopper with Low Friction). **(2) Interleaved Periodic:** This pattern injects noise slots between recurring motifs to stress-test memory robustness. In **XLand-Minigrid**, a specific rule motif repeats at fixed intervals, separated by blocks of randomly sampled rules. Similarly, in **Kinetix**, a structured configuration recurs between blocks of independently sampled configurations from the same held-out structured task pool.

(2) Linear Drift. We implement gradual change through domain-specific sequences of task configurations. In **XLand-Minigrid**, this takes the form of a **Complexity Curriculum**, with rule depth and the number of distractors increasing progressively across task blocks. This tests the agent's adaptation as symbolic reasoning becomes more demanding. In **Kinetix**, morphology remains fixed

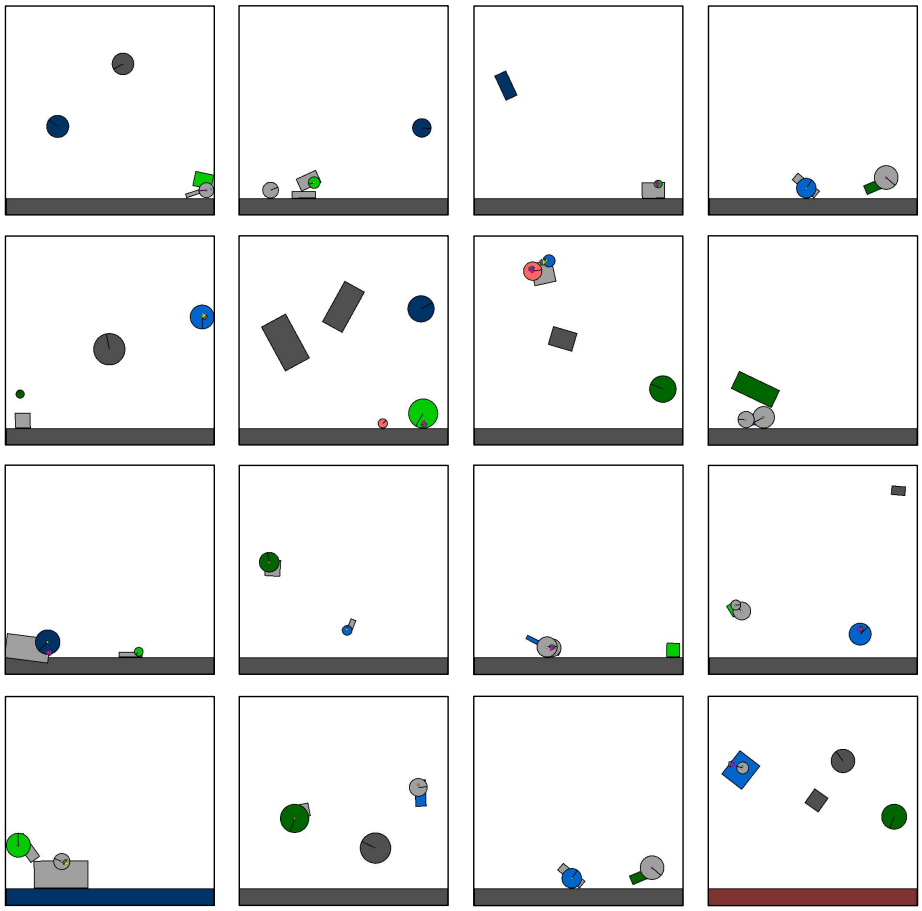


Figure 7: **Kinetix Training Environment.** An example of a procedurally generated, unstructured level used during training. The agent controls motors and thrusters to manipulate random morphologies.

Table 9: **Kinetix Physics Parameter Space.** Parameters are sampled from Uniform (U) or Log-Uniform ($LogU$) distributions to cover a wide dynamic range.

Scope	Parameter	Sampling Distribution
Global	Gravity Vector Magnitude	$U(0.0, 20.0)$
	Base Thruster Force	$LogU(\ln 0.2, \ln 5.0) \times 10.0$
	Base Motor Torque	$LogU(\ln 0.2, \ln 2.0) \times 900.0$
	Global Joint Stiffness	$U(0.2, 1.0)$
Local	Object Density	$LogU(\ln 0.2, \ln 5.0)$
	Friction Coefficient	$U(0.1, 2.0)$
	Restitution (Elasticity)	$U(0.0, 0.6)$

while physics parameters evolve as $\Phi_b = (1 - \alpha_b)\Phi_{\text{start}} + \alpha_b \Phi_{\text{end}}$, where b indexes task blocks and α_b increases from 0 to 1 across the lifetime.

(3) Noisy Fluctuation. The configuration varies around a fixed base configuration according to $\mathcal{E}_b \sim \mathcal{U}_{\text{loc}}(\mathcal{E}_{\text{base}})$, where b indexes ten-episode blocks and $\mathcal{U}_{\text{loc}}$ specifies the domain-specific perturbations. In **XLand-Minigrid**, the rule graph's causal structure remains fixed, while a consistent permutation of object colors and shapes is applied to the rules, goals, and initial objects at each block boundary. In **Kinetix**, morphology remains fixed and the physics parameters are generated as $\Phi_b = (1 -$

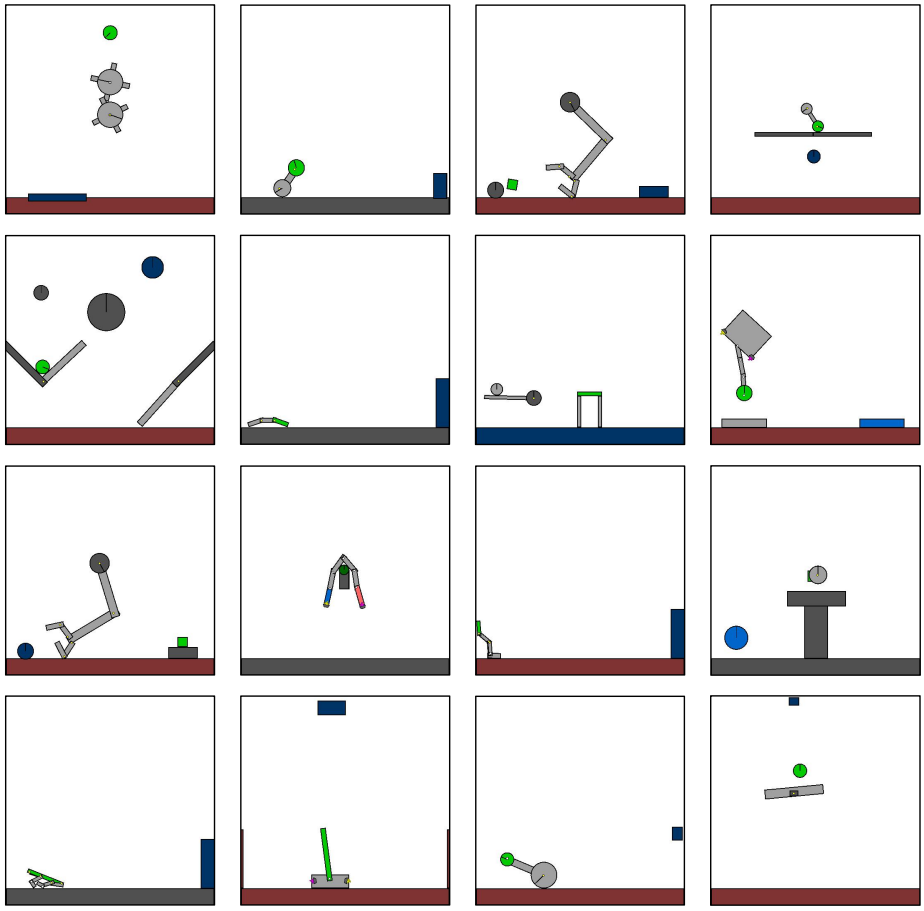


Figure 8: **Kinetix Test Environment.** An example of a structured evaluation level (e.g., a Walker task). Even in these structured tasks, the underlying physics (gravity, friction) change dynamically.

$\eta)\Phi_{\text{base}} + \eta\widetilde{\Phi}_b$, where $\widetilde{\Phi}_b$ is an independently randomized parameter vector and $\eta = 0.2$ controls the perturbation strength.

B Experimental Setup Details

To rigorously evaluate In-Context Reinforcement Learning (ICRL) capabilities across different architectures, we employ a unified policy network framework. The model consists of a domain-specific `Time-step Encoder`, a `Sequence Modeling Backbone`, and shared `Actor-Critic Heads`. To ensure a fair comparison of model capacity, all sequence backbones are standardized to a hidden dimension of $d_{model} = 512$ and a depth of $L = 6$ layers.

B.1 Evaluation Setup

An evaluation lifetime consists of $K = 200$ episodes in XLand-Minigrid or $K = 100$ episodes in Kinetix. The default evaluation uses 4096 parallel environments for XLand-Minigrid and 2048 for Kinetix, with returns averaged across environments. The comparison with online RL methods (Table 6) uses 32 independent single-environment runs per method (seeds 1–32), including Frozen LICRL. The specific configurations for each protocol are as follows:

- **Protocol I (Stationary):** The environment configuration remains fixed throughout the lifetime.

- **Protocol II (Random Switching):** We evaluate two distinct volatility levels:
  - **Fast Switching:** The configuration is resampled every 1 episode.
  - **Slow Switching:** The configuration is resampled every 10 episodes.
- **Protocol III (Structured Non-Stationarity):** For all structured patterns (Periodic Recurrence, Linear Drift, Noisy Fluctuation), the underlying dynamics evolve or switch every 10 episodes, balancing the need for stability to allow adaptation with sufficient transitions to test flexibility.

B.2 Input Representations and Encoders

At each time step t, the model input is a tuple $x_t = (o_t, a_{t-1}, r_{t-1})$, containing the current observation, the previous action, and the previous reward. The encoder ϕ processes these into a unified embedding $e_t \in \mathbb{R}^{d_{model}}$.

XLand-Minigrid (Symbolic & Discrete). Observations are $H \times W \times 2$ symbolic grids consisting of object IDs and color IDs. We first map these discrete indices to learned embeddings, which are concatenated and processed by a 3-layer Convolutional Neural Network (CNN) without pooling to preserve spatial information. The architecture consists of: (1) 16 filters with 2×2 kernels; (2) 32 filters with 2×2 kernels; and (3) 64 filters with 2×2 kernels. All layers use ReLU activations and a stride of 1. To integrate context, the previous discrete action a_{t-1} is mapped via a learned lookup table, while the scalar reward r_{t-1} is projected via a linear layer. These embeddings are concatenated with the flattened visual features and fused via a linear projection to form the model input.

Kinetix (Pixel & Continuous). Observations are high-dimensional RGB pixel arrays. We employ a 3-layer CNN architecture similar to standard Atari encoders but adapted for physics-based control: (1) 32 filters with 8×8 kernels and stride 4; (2) 64 filters with 4×4 kernels and stride 2; and (3) 64 filters with 3×3 kernels and stride 2. All layers use ReLU activations. Since Kinetix features a continuous action space, the previous action vector a_{t-1} is encoded via a linear layer rather than an embedding table. The resulting visual, action, and reward embeddings are concatenated and linearly projected to d_{model}.

Table 10: **Detailed Hyperparameters for XLand-Minigrid and Kinetix.**

Hyperparameter	XLand-Minigrid	Kinetix
Optimization & PPO		
Total Timesteps	1×10^{10} (10B)	1×10^{9} (1B)
Optimizer	Adam	
Learning Rate	1×10^{-3}	3×10^{-4}
Num. Environments	8,192	4,096
Discount Factor (γ)	0.99	
GAE Smoothing (λ)	0.95	
Clip Range (ϵ)	0.2	
Entropy Coeff.	0.01	
Value Coeff.	0.5	
Update Epochs	2	
Num. Minibatches	256	
Training Context & TBPTT		
Stationary Training		
Max Context Length	8,192	4,096
BPTT Horizon	2,048	1,024
Non-Stationary Training		
Max Context Length	16,384	8,192
BPTT Horizon	2,048	1,024
Switch Interval	Rand $\in [1, 80]$	Rand $\in [1, 80]$

B.3 Sequence Modeling Backbones

We evaluate four distinct architectures representing different inductive biases for memory and computation. All models utilize $L = 6$ layers and a model dimension of $d_{model} = 512$.

Sliding Window Attention. To manage the computational cost of long lifetimes, we employ a Decoder-only Transformer with **Sliding Window Attention**. The window size is set to $W = 512$, limiting the receptive field to the most recent transitions. The model is configured with 8 attention heads (head dimension 64). During inference, this architecture utilizes a Key-Value (KV) cache with memory complexity scaling linearly with W rather than the total lifetime length, enforcing a strong bias towards local history.

Gated DeltaNet (Linear Attention). We adopt Gated DeltaNet [24] as our primary linear attention baseline. This architecture combines the efficiency of Linear Attention with the gating mechanisms of RNNs. It maintains a fixed-size state $S_t \in \mathbb{R}^{d_{model} \times d_{head}}$ updateable via outer products of keys and values. Crucially, it employs a data-dependent decay gate α_t that allows the model to selectively forget historical information based on current inputs—a vital feature for non-stationary environments where old rules must be discarded. The model uses 6 heads with a head dimension of 64 and enables $O(1)$ inference and $O(T)$ training complexity.

Mamba2 (SSM). We evaluate Mamba2 [21], a recent advancement in State Space Models (SSMs). Mamba2 simplifies the SSM parameterization using the Structured State Space duality (SSD), allowing it to be viewed as a constrained form of linear attention. We use a state expansion factor of $d_{state} = 64$ and a convolutional kernel size of 4 for the local mixing layer.

Hybrid Architecture. To leverage both the precise recall of attention and the infinite context summary of recurrent models, we introduce a Hybrid architecture. This model consists of 6 layers primarily based on Gated DeltaNet, but with the 3rd and 6th layers replaced by Sliding Window Attention layers ($W = 512$, 8 heads).

B.4 Training Hyperparameters

We train all agents using Proximal Policy Optimization (PPO) with Truncated Backpropagation Through Time (TBPTT). The training infrastructure is implemented in PyTorch, using the Flash Linear Attention (FLA) library for efficient sequence operations. Table 10 details the domain-specific configurations. Note that for Non-Stationary regimes, we double the context length to ensure the agent experiences dynamic shifts within a single backpropagation window.

B.5 Compute Resources.

All experiments were conducted on machines equipped with NVIDIA A800 (80GB) GPUs. A single 10B-step XLand-Minigrid training run takes approximately 21 hours on one GPU, and a 1B-step Kinetix run requires approximately 20 hours.

C Additional Experimental Results

C.1 Dimension 1: Training Regimes

We assess the robustness of the training-regime results using five training seeds (41–45), including seed 42 used in Table 1. Table 11 reports the results for Stationary, Non-Stationary (NS), and Composite Non-Stationary (CNS) training with Gated DeltaNet.

We use paired two-sided t-tests and apply Holm–Bonferroni correction at $\alpha = 0.05$ across 15 comparisons. These comprise all three regime pairs on the four protocol-specific R_{Seq} metrics and Random Switch Δ_{Switch}. Both NS and CNS outperform Stationary training on Random Switch and Periodic R_{Seq}, as well as Random Switch Δ_{Switch}, with all six comparisons remaining significant after correction. CNS has the highest mean R_{Seq} under all four protocols, although none of its comparisons with NS remains significant after correction.

Table 11: Performance of training regimes across XLand evaluation protocols (mean $\pm$ standard deviation over 5 training seeds). Bold indicates the highest mean in each column.

Training regime	Stationary	Random Switch		Periodic	Drift
	R_{Seq}	R_{Seq}	Δ_{Switch}	R_{Seq}	R_{Seq}
Stationary	0.150 ± 0.010	0.057 ± 0.022	-0.101 ± 0.023	0.065 ± 0.012	0.206 ± 0.006
NS	0.167 ± 0.009	0.154 ± 0.010	$\mathbf{-0.019} \pm 0.004$	0.113 ± 0.009	0.202 ± 0.010
CNS	$\mathbf{0.175} \pm 0.008$	$\mathbf{0.155} \pm 0.004$	-0.027 ± 0.006	$\mathbf{0.119} \pm 0.005$	$\mathbf{0.214} \pm 0.009$

Table 12: Detailed performance metrics of different training paradigms evaluated on the XLand-Minigrid **Stationary** protocol. We report the Zero-Shot baseline (R_{ZS}), followed by Mean Returns (R) and Improvement scores (Δ) across History, Switch, and Stable phases.

Training Regime	Base	History (Seq)		Switch Phase		Stable Phase	
	R_{ZS}	R_{Seq}	Δ_{Ctx}	R_{Switch}	Δ_{Switch}	R_{Stable}	Δ_{Static}
(1) Domain Randomization	0.149	0.128	-0.021	0.149	0.0	0.128	-0.021
(2) Stationary	0.158	0.154	-0.003	0.158	0.0	0.154	-0.003
(3) Non-Stationary	0.151	0.165	0.014	0.151	0.0	0.165	0.014
(4) Composite Non-Stationary	**0.164**	**0.181**	**0.017**	**0.164**	0.0	**0.182**	**0.017**
(5) Curr: DR $\rightarrow$ Stat	0.126	0.119	-0.007	0.126	0.0	0.119	-0.007
(6) Curr: DR $\rightarrow$ Non-Stat	0.141	0.150	0.009	0.141	0.0	0.150	0.009

This appendix provides a more comprehensive quantitative analysis of Training Regimes, utilizing fine-grained metrics (such as performance during switch phases Δ_{Switch} and stable phases Δ_{Static}) to further elucidate the impact of different training strategies on lifelong adaptation capabilities.

First, under the Stationary evaluation protocol, CNS achieves a higher sequence return than Stationary training ($R_{\text{Seq}} = 0.181$ versus 0.154, Table 12 and Figure 9). When tasks switch every 10 episodes (Table 13 and Figure 10), the Stationary-trained agent exhibits a large negative switch gain ($\Delta_{\text{Switch}} = -0.094$), compared with -0.027 for NS and -0.028 for CNS. These results suggest that non-stationary training can improve robustness to task changes while maintaining effective performance when the task is fixed.

Second, the advantage of non-stationary training persists when tasks switch every episode. As shown in Table 14 and Figure 11, NS and CNS achieve sequence returns of 0.137 and 0.139, respectively, compared with 0.021 for Stationary training and 0.034 for DR. The sequence returns of NS and CNS remain closer to their respective zero-shot baselines, suggesting reduced interference from outdated context even without stable multi-episode blocks.

Finally, under Linear Drift (Table 17), CNS achieves the largest context gain among the six training regimes ($\Delta_{\text{Ctx}} = 0.073$), compared with 0.054 for Stationary training. Under Noisy Fluctuation (Table 18), CNS again achieves the largest context gain ($\Delta_{\text{Ctx}} = 0.011$), while NS and DR $\rightarrow$ NS also yield positive gains (0.002 and 0.003, respectively). Since CNS training combines Random Switch and Periodic Recurrence, these results suggest that its adaptation benefits extend to unseen forms of structured non-stationarity.

C.1.1 Evaluation on Matched Tasks

CNS obtains a lower XLand sequence return under Periodic Recurrence than under Random Switch in Table 1. This gap reflects differences in task difficulty between the independently sampled task batches used for the two protocols. To compare the two protocols while controlling for task difficulty, we evaluate the same CNS model on a shared batch of held-out XLand tasks (Table 19).

The similar zero-shot returns indicate comparable task difficulty, and CNS achieves positive context gains under both protocols. In this matched comparison, Periodic Recurrence yields a higher sequence return and a larger context gain, consistent with the agent benefiting from recurring task structure.

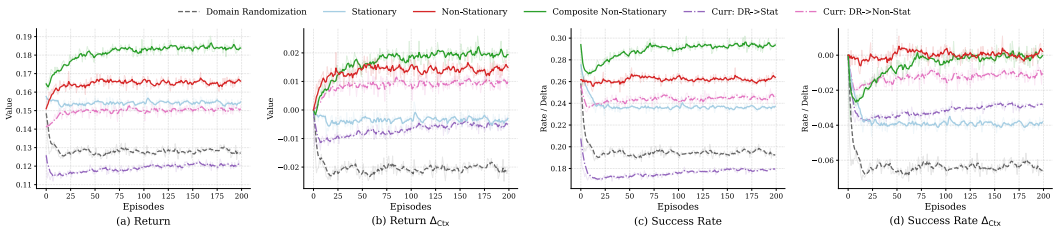


Figure 9: Results of Different Training Regimes under the Stationary Testing Protocol in XLand-Minigrid

Table 13: Detailed performance metrics on the XLand-Minigrid **Non-Stationary** protocol, where tasks switch randomly every 10 episodes.

Training Regime	**Base**	**History (Seq)**		**Switch Phase**		**Stable Phase**	
	R_{ZS}	R_{Seq}	Δ_{Ctx}	R_{Switch}	Δ_{Switch}	R_{Stable}	Δ_{Static}
(1) Domain Randomization	0.145	0.100	-0.045	0.051	-0.094	0.106	-0.040
(2) Stationary	0.150	0.074	-0.077	0.056	-0.094	0.076	-0.075
(3) Non-Stationary	0.146	0.153	0.007	0.119	-0.027	0.157	0.011
(4) Composite Non-Stationary	**0.156**	**0.167**	**0.010**	**0.128**	-0.028	**0.171**	**0.015**
(5) Curr: DR $\rightarrow$ Stat	0.119	0.044	-0.075	0.028	-0.091	0.046	-0.074
(6) Curr: DR $\rightarrow$ Non-Stat	0.135	0.141	0.006	0.111	**-0.023**	0.144	0.010

C.1.2 Hidden State Dynamics

We assess the robustness of the layer-wise analysis in Section 5.3 using five training seeds for Stationary, Non-Stationary (NS), and Composite Non-Stationary (CNS) training. For each layer, Δ is the mean hidden-state cosine similarity during stable periods minus that at task switches. A positive Δ indicates that hidden states change more at task switches than during stable periods. Table 20 reports the results for the middle and top layers.

Stationary training yields a near-zero shift at L5, while NS and CNS yield positive shifts at both L3 and L5. This pattern is consistent with more pronounced state updates at task switches after non-stationary training.

C.2 Dimension 2: Model Architectures

We assess the robustness of the XLand architecture comparison using five training seeds for Gated DeltaNet, Sliding Window Attention (SWA), and Hybrid under the Non-Stationary (NS) training regime. Table 21 reports sequence returns across the XLand evaluation protocols.

Gated DeltaNet and Hybrid outperform SWA under every evaluation protocol ($p < 0.0001$ for both comparisons under each protocol). Hybrid achieves the highest mean R_{Seq} among the three architectures across all four protocols.

C.3 Robustness Across Backbones and Training Regimes

This supplementary experiment tests whether the findings from the main comparisons depend on the fixed backbone or training regime. We replace Gated DeltaNet with Hybrid when comparing Stationary and Composite Non-Stationary (CNS) training. We also replace Non-Stationary (NS) with CNS as the common training regime when comparing Hybrid and sliding-window attention (SWA). Table 22 reports XLand sequence returns as means $\pm$ standard deviations over five training seeds.

With Hybrid, CNS achieves higher mean sequence returns than Stationary training across all four protocols. Under CNS training, Hybrid achieves higher mean sequence returns than SWA across all four protocols.

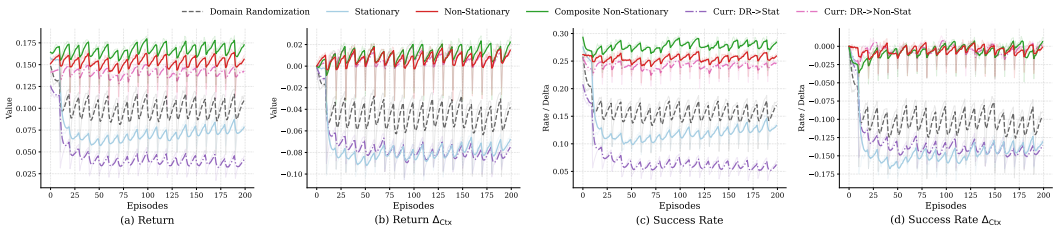


Figure 10: Results of Different Training Regimes under the Non-Stationary (10) Testing Protocol in XLand-Minigrid

Table 14: Detailed performance metrics on the XLand-Minigrid **Non-Stationary** protocol, where tasks switch randomly every 1 episode.

Training Regime	**Base**	**History (Seq)**		**Switch Phase**		**Stable Phase**	
	R_{ZS}	R_{Seq}	Δ_{Ctx}	R_{Switch}	Δ_{Switch}	R_{Stable}	Δ_{Static}
(1) Domain Randomization	0.146	0.034	-0.112	0.034	-0.112	0.0	0.0
(2) Stationary	0.150	0.021	-0.129	0.021	-0.129	0.0	0.0
(3) Non-Stationary	0.146	0.137	**-0.009**	0.137	**-0.009**	0.0	0.0
(4) Composite Non-Stationary	**0.156**	**0.139**	-0.017	**0.139**	-0.017	0.0	0.0
(5) Curr: DR $\rightarrow$ Stat	0.119	0.016	-0.104	0.016	-0.104	0.0	0.0
(6) Curr: DR $\rightarrow$ Non-Stat	0.134	0.123	-0.011	0.123	-0.011	0.0	0.0

D Limitations and Broader Impacts

Limitations. Our evaluation is conducted in two simulated domains under controlled non-stationary protocols. While these settings capture representative forms of intra-lifetime dynamics, real-world environments may involve more complex and less predictable changes that are not fully covered here. Furthermore, our current analysis is empirical in nature, and a formal theoretical characterization of how frozen sequence models perform implicit context inference is left for future investigation.

Broader Impacts. This work presents a systematic empirical study of lifelong in-context adaptation in simulated environments. As foundational research, we foresee no direct negative societal consequences.

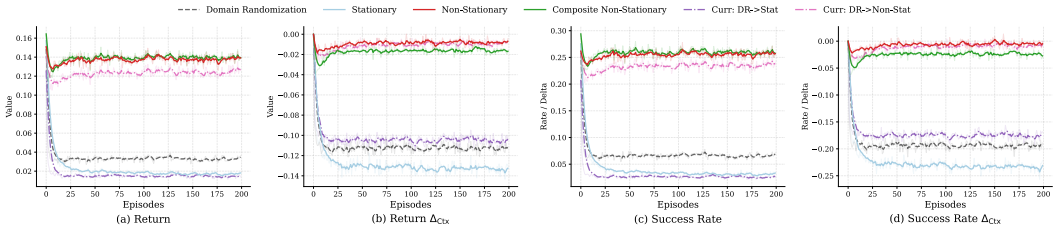


Figure 11: Results of Different Training Regimes under the Non-Stationary (1) Testing Protocol in XLand-Minigrid

Table 15: Detailed performance metrics on the XLand-Minigrid **Standard Periodic** protocol, where tasks switch every 10 episodes.

Training Regime	**Base**	**History (Seq)**		**Switch Phase**		**Stable Phase**	
	R_{ZS}	R_{Seq}	Δ_{Ctx}	R_{Switch}	Δ_{Switch}	R_{Stable}	Δ_{Static}
(1) Domain Randomization	0.102	0.056	-0.046	0.026	-0.076	0.059	-0.043
(2) Stationary	0.108	0.063	-0.046	0.058	-0.050	0.063	-0.045
(3) Non-Stationary	0.104	0.103	-0.001	0.081	-0.023	0.105	0.001
(4) Composite Non-Stationary	**0.112**	**0.115**	**0.003**	**0.093**	**-0.019**	**0.117**	**0.006**
(5) Curr: DR $\rightarrow$ Stat	0.079	0.037	-0.041	0.033	-0.046	0.038	-0.041
(6) Curr: DR $\rightarrow$ Non-Stat	0.093	0.094	0.001	0.074	**-0.019**	0.096	0.004

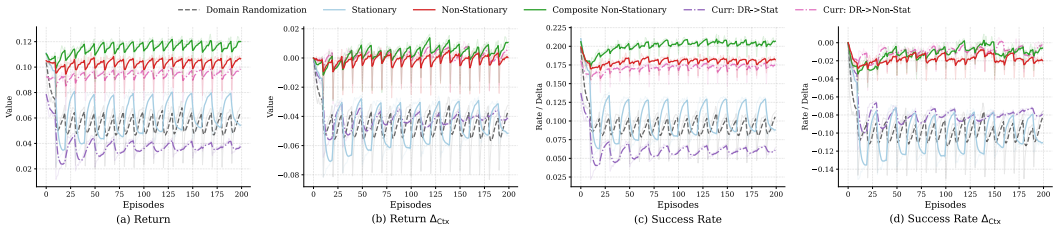


Figure 12: Results of Different Training Regimes under the Standard Periodic Testing Protocol in XLand-Minigrid

Table 16: Detailed performance metrics on the XLand-Minigrid **Interleaved Periodic** protocol, where tasks switch every 10 episodes.

Training Regime	**Base**	**History (Seq)**		**Switch Phase**		**Stable Phase**	
	R_{ZS}	R_{Seq}	Δ_{Ctx}	R_{Switch}	Δ_{Switch}	R_{Stable}	Δ_{Static}
(1) Domain Randomization	0.106	0.059	-0.046	0.027	-0.078	0.063	-0.043
(2) Stationary	0.112	0.058	-0.054	0.053	-0.059	0.059	-0.053
(3) Non-Stationary	0.106	0.105	-0.001	0.083	-0.023	0.108	0.002
(4) Composite Non-Stationary	**0.116**	**0.118**	**0.002**	**0.093**	-0.023	**0.121**	**0.005**
(5) Curr: DR $\rightarrow$ Stat	0.084	0.035	-0.049	0.029	-0.055	0.035	-0.048
(6) Curr: DR $\rightarrow$ Non-Stat	0.097	0.097	0.001	0.077	**-0.020**	0.100	0.003

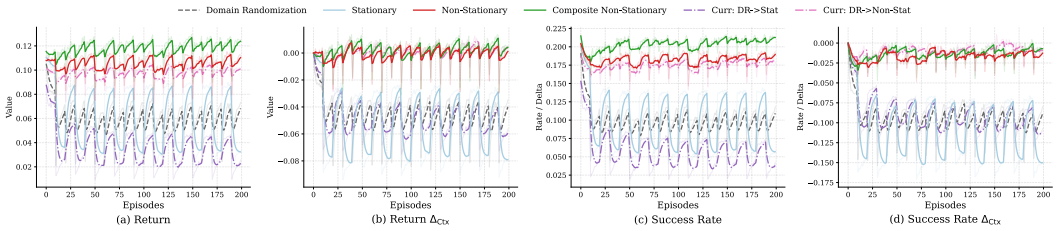


Figure 13: Results of Different Training Regimes under the Interleaved Periodic Testing Protocol in XLand-Minigrid

Table 17: Detailed performance metrics on the XLand-Minigrid **Linear Drift** protocol, where tasks switch every 10 episodes.

Training Regime	**Base**	**History (Seq)**		**Switch Phase**		**Stable Phase**	
	R_{ZS}	R_{Seq}	Δ_{Ctx}	R_{Switch}	Δ_{Switch}	R_{Stable}	Δ_{Static}
(1) Domain Randomization	0.234	0.242	0.008	0.239	0.005	0.243	0.009
(2) Stationary	**0.251**	0.305	0.054	**0.293**	0.042	0.306	0.055
(3) Non-Stationary	0.237	0.290	0.053	0.280	0.042	0.291	0.054
(4) Composite Non-Stationary	0.233	**0.306**	**0.073**	0.288	**0.055**	**0.308**	**0.075**
(5) Curr: DR $\rightarrow$ Stat	0.224	0.262	0.038	0.258	0.035	0.262	0.039
(6) Curr: DR $\rightarrow$ Non-Stat	0.230	0.268	0.038	0.264	0.034	0.269	0.038

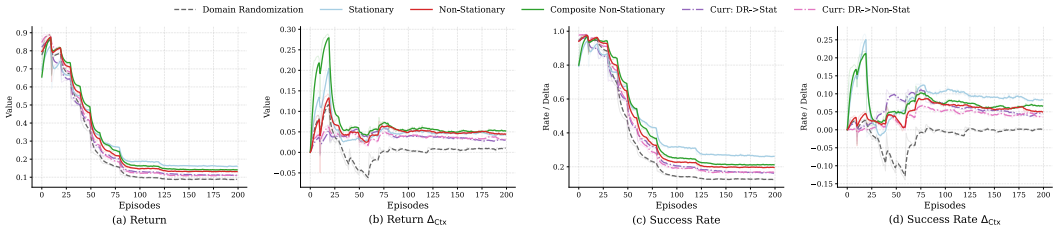


Figure 14: Results of Different Training Regimes under the Linear Drift Testing Protocol in XLand-Minigrid

Table 18: Detailed performance metrics on the XLand-Minigrid **Noisy Fluctuation** protocol, where tasks switch every 10 episodes.

Training Regime	**Base**	**History (Seq)**		**Switch Phase**		**Stable Phase**	
	R_{ZS}	R_{Seq}	Δ_{Ctx}	R_{Switch}	Δ_{Switch}	R_{Stable}	Δ_{Static}
(1) Domain Randomization	0.102	0.076	-0.026	0.077	-0.025	0.076	-0.026
(2) Stationary	0.111	0.096	-0.015	0.097	-0.014	0.095	-0.016
(3) Non-Stationary	0.105	0.107	0.002	0.108	0.003	0.106	0.001
(4) Composite Non-Stationary	**0.112**	**0.122**	**0.011**	**0.121**	**0.009**	**0.122**	**0.011**
(5) Curr: DR $\rightarrow$ Stat	0.080	0.070	-0.010	0.071	-0.008	0.069	-0.010
(6) Curr: DR $\rightarrow$ Non-Stat	0.094	0.097	0.003	0.097	0.003	0.097	0.003

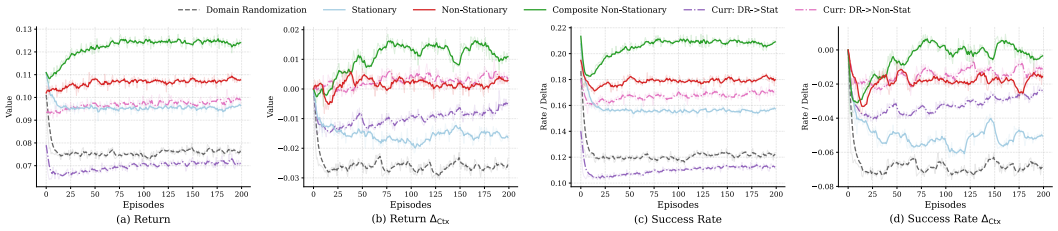


Figure 15: Results of Different Training Regimes under the Noisy Fluctuation Testing Protocol in XLand-Minigrid

Table 19: CNS evaluated on matched XLand tasks.

Evaluation protocol	R_{ZS}	R_{Seq}	Δ_{Ctx}
Random Switch	0.156	0.168	+0.012
Periodic Recurrence	0.157	0.171	+0.014

Table 20: Five-seed results for hidden-state similarity shifts Δ (mean $\pm$ standard deviation).

Layer	**Training regime**	Δ
L3 (Middle)	NS	$+0.115 \pm 0.028$
L3 (Middle)	CNS	$+0.102 \pm 0.024$
L5 (Top)	Stationary	$+0.004 \pm 0.006$
L5 (Top)	NS	$+0.095 \pm 0.053$
L5 (Top)	CNS	$+0.122 \pm 0.051$

Table 21: Sequence returns across XLand evaluation protocols under NS training (mean $\pm$ standard deviation over five training seeds). Bold indicates the highest mean in each column.

Architecture	**Stationary**	**Random Switch**	**Periodic**	**Drift**
Gated DeltaNet	0.167 ± 0.009	0.154 ± 0.010	0.113 ± 0.009	0.202 ± 0.010
Hybrid	$\mathbf{0.173} \pm 0.004$	$\mathbf{0.158} \pm 0.006$	$\mathbf{0.116} \pm 0.005$	$\mathbf{0.207} \pm 0.006$
SWA	0.076 ± 0.004	0.053 ± 0.004	0.044 ± 0.005	0.101 ± 0.002

Table 22: Sequence returns across training regimes and backbones in XLand.

Backbone	**Training regime**	**Stationary**	**Random Switch**	**Periodic**	**Drift**
Hybrid	Stationary	0.144 ± 0.011	0.069 ± 0.002	0.064 ± 0.001	0.211 ± 0.015
Hybrid	CNS	0.171 ± 0.006	0.162 ± 0.002	0.120 ± 0.008	0.218 ± 0.011
SWA	CNS	0.094 ± 0.007	0.073 ± 0.003	0.051 ± 0.005	0.135 ± 0.005

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the main contributions.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses limitations in Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: This paper is primarily empirical and does not contain theorems or proofs requiring theoretical assumptions.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All reproduction details, including architectures, hyperparameters, and non-stationary protocols, are specified in Section 5.1 and Appendices A, B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The codebase is currently being cleaned for release and will be made publicly available upon acceptance.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Training and evaluation details, including hyperparameters and protocols, are provided in Section 5.1 and Appendices A, B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Due to 10B-step compute costs, we report single runs but evaluate across 200 episodes and 1024 OOD task sequences to ensure representative performance.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the compute resources used in App. B.5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and confirm that this work conforms with it in every respect.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: As foundational research in simulated environments, direct negative impacts are not foreseen; a brief discussion is provided in Appendix D.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our work focuses on RL agents in simulated control tasks, posing no significant risk of misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All assets (XLand-Minigrid, Kinetix, Mamba2, Gated DeltaNet) are cited and used under their respective Apache 2.0 and MIT open-source licenses.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new evaluation suite is documented in the paper and appendix; the codebase will be released upon acceptance.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve any crowdsourcing or research conducted with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: All experiments are performed in simulated environments without the participation of human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were used only for minor writing and editing assistance.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.