Online Reinforcement Learning in Non-Stationary Context-Driven Environments

Anonymous authors
Paper under double-blind review

Abstract

We study online reinforcement learning (RL) in non-stationary environments, where a time-varying exogenous context process affects the environment dynamics. Online RL is challenging in such environments due to "catastrophic forgetting" (CF). The agent tends to forget prior knowledge as it trains on new experiences. Prior approaches to mitigate this issue assume task labels (which are often not available in practice), employ brittle regularization heuristics or use off-policy methods that suffer from instability and poor performance.

We present Locally Constrained Policy Optimization (LCPO), an online RL approach that combats CF by anchoring policy outputs on old experiences while optimizing the return on current experiences. To perform this anchoring, LCPO locally constrains policy optimization using samples from experiences that lie outside of the current context distribution. We evaluate LCPO in Mujoco, classic control and computer systems environments with a variety of synthetic and real context traces, and find that it outperforms a variety of baselines in the non-stationary setting, while achieving results on-par with a "prescient" agent trained offline across all context traces.

1 Introduction

— *Those who cannot remember the past are condemned to repeat it. (George Santayana, The Life of Reason, 1905)*

Reinforcement Learning (RL) has seen success in many domains (Mao et al., 2017; Haarnoja et al., 2018a; Mao et al., 2019b; Marcus et al., 2019; Zhu et al., 2020; Haydari & Yılmaz, 2022), but real-world deployments have been rare. A major hurdle has been the gap between simulation and reality, where the environment simulators do not match the real-world dynamics. Thus, recent work has turned to applying RL in an *online* fashion, i.e. continuously training and using an agent in a live environment (Zhang et al., 2021; Gu et al., 2021).

While online RL is difficult in and of itself, it is particularly challenging in *non-stationary* environments (also known as continual RL (Khetarpal et al., 2020)), where the characteristics of the environment change over time. A key challenge is Catastrophic Forgetting (CF) (McCloskey & Cohen, 1989). An agent based on function approximators like Neural Networks (NNs) tends to forget its past knowledge when training sequentially on new non-stationary data. On-policy RL algorithms (Sutton & Barto, 2018) are particularly vulnerable to CF in non-stationary environments, since these methods cannot retrain on stale data from prior experiences.

In this paper, we consider problems where the source of the non-stationarity is an observed exogenous *context* process that varies over time and exposes the agent to different environment dynamics. Such context-driven environments (Sinclair et al., 2023; Mao et al., 2018; Zhang et al., 2023; Dietterich et al., 2018; Pan et al., 2022) appear in a variety of applications. Examples include computer systems subject to incoming workloads (Mao et al., 2018), locomotion in environments with varying terrains and obstacles (Heess et al., 2017b), robots subject to external forces (Pinto et al., 2017), and more. In contrast to most prior work (Alegre et al., 2021; Chandak et al., 2020), we do not restrict the context process to be discrete, piece-wise stationary or Markov.

Broadly speaking, there are three existing approaches to mitigate CF in online learning. One class of techniques are *task-based* (Rusu et al., 2016; Kirkpatrick et al., 2017; Schwarz et al., 2018; Farajtabar et al., 2019; Zeng et al., 2019). They assume explicit task labels that identify the different context distributions which the agent encounters over time. Task labels make

it easier to prevent the training for one context from affecting knowledge learned for other contexts. In settings where task labels (or boundaries) are not available, a few proposals try to infer the task labels via self-supervised (Nagabandi et al., 2019b) or Change-Point Detection (CPD) approaches (Padakandla et al., 2020; Alegre et al., 2021). These techniques, however, are brittle when the context processes are difficult to separate and task boundaries are not pronounced (Hamadanian et al., 2022). Our experiments show that erroneous task labels lead to poorly performing agents in such environments (§5).

A second category of approaches avoid task labels by approximating task-based methods with heuristics (Schwarz et al., 2018; Chaudhry et al., 2018; Kaplanis et al., 2018; Woo et al., 2022). However, these heuristics are based on brittle assumptions about the nature and cadence of non-stationarity. For example, one approach implicitly assumes each episode is a distinct task, and uses a window of past N episode to regularize learning (Woo et al., 2022). These assumptions are rarely met and would likely lead to poor performance in practice, as we observe in our analysis and evaluations(§5 and §D.1, §D.2 and §D.3 in the Appendix).

The third category of approaches employ rehearsal, i.e., learning using past or generated data. For example, off-policy learning (Sutton & Barto, 2018) makes it possible to retrain on past data. These techniques (e.g., Experience Replay (Mnih et al., 2013), CLEAR (Rolnick et al., 2019), etc.) store prior experience data in a buffer and sample from the buffer randomly to train. Not only does this improve sample complexity, it sidesteps the pitfalls of sequential learning and prevents CF (Rolnick et al., 2019). However, off-policy methods come at the cost of increased hyper-parameter sensitivity and unstable training (Duan et al., 2016; Gu et al., 2016; Haarnoja et al., 2018b). This brittleness is particularly catastrophic in an online setting, as we also observe in our experiments (§5).

We present LCPO (§4.1), an on-policy RL algorithm that "anchors" policy outputs for old contexts while optimizing for the current context. Unlike prior work, LCPO does not rely on task labels and only requires an Out-of-Distribution (OOD) detector, i.e., a function that recognizes old experiences that occurred in a sufficiently different context than the current one. LCPO maintains a bounded buffer of past experiences, similar to off-policy methods (Mnih et al., 2013). But as an on-policy approach, LCPO does not use stale experiences to optimize the policy. Instead, it uses past data to *constrain* the policy optimization on fresh data, such that the agent's behavior does not change in other contexts.

We evaluate LCPO on several environments with real and synthetic contexts (§5), and show that it outperforms a variety of baselines across mentioned categories in the online learning setting. We also compare against a "prescient agent" that is trained offline on the entire context distribution prior to deployment. The prescient agent does not suffer from CF. Among all the online methods, LCPO is the closest to this idealized baseline. Our ablation results show that LCPO is robust to variations in the OOD detector's thresholds and works well with small experience buffer sizes. LCPO's source code is available online at `https://github.com/LCPO-RL/LCPO`.

2 Preliminaries

Notation. We consider online reinforcement learning in a non-stationary context-driven Markov Decision Process (MDP), where the context is observed (only up to the current time step t) and exogenous. Formally, at time step t the environment has state $s_t \in \mathcal{S}$ and context $z_t \in \mathcal{Z}$. The agent takes action $a_t \in \mathcal{A}$ based on the observed state s_t and context z_t, $a_t = \pi(s_t, z_t)$, and receives feedback in the form of a scalar reward $r_t = r(s_t, z_t, a_t)$, where $r(\cdot,\cdot,\cdot) : \mathcal{S} \times \mathcal{Z} \times \mathcal{A} \to \mathbb{R}$ is the reward function. The environment's state, the context, and the agent's action determine the next state, s_{t+1}, according to a transition kernel, $T(s_{t+1}|s_t, z_t, a_t)$. The context z_t is an independent stochastic process, unaffected by states s_t or actions a_t. Finally, d_0 defines the distribution over initial states (s_0). This model is fully defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{Z}, \mathcal{A}, \{z_t\}_{t=1}^{\infty}, T, d_0, r)$.

Non-stationary contexts. The non-stationary context $\mathbf{z} = \{z_t\}_{t=1}^{\infty}$ impacts the environment dynamics and implies a non-stationary environment. We assume the context process can change arbitrarily: e.g., it can follow a predictable pattern, be i.i.d samples from some distribution, be a discrete process or a multi-dimensional continuous process, experience smooth or dramatic shifts, be piece-wise stationary, or include any mixture of the above. We have no prior knowledge of the context process, the environment dynamics, or access to an offline simulator. Examples of (observed) context processes include market demand in a supply chain system, incoming request workloads in virtual machine allocation problem, customer distributions in

airline revenue management (Sinclair et al., 2023), traffic information in vehicular networks (Wu et al., 2017), terrain profiles in a locomotion task (Heess et al., 2017a), network traffic for video streaming (Mao et al., 2020) and congestion control (Winstein & Balakrishnan, 2013), etc.

Goal. We seek good long-term performance. Formally, for a given policy $\pi:\mathcal{S}\times\mathcal{Z}\rightarrow\mathcal{A}$ and context process $\mathbf{z}=\{z_t\}_{t=1}^{\infty}$ we define the lifelong return as $J(\pi,\mathbf{z})=\lim_{t\rightarrow\infty}\sum_{i=1}^{t}\frac{r_i}{t}$ for an infinite horizon MDP. Similarly, for finite horizon MDPs of length H where episode i is subject to context traces $\mathbf{z}_i=(z_{H.i},z_{H.i+1},...,z_{H.(i+1)-1})$ and has an episodic return of $R_i=\sum_{t=1}^{H}r_t^{(i)}$, we define the lifelong return as $J(\pi,\mathbf{z})=\lim_{t\rightarrow\infty}\sum_{i=1}^{t}\frac{R_i}{t}$. For a policy sequence $\mathbf{\Pi}=\{\pi_t\}_{t=1}^{\infty}$, e.g., the sequence of policies resulting from a continual RL algorithm, we can define the lifelong return $J(\mathbf{\Pi},\mathbf{z})$ similarly.

Online RL. In most RL settings, a policy is trained in a separate training phase. During testing, the policy is fixed and does not change. By contrast, online learning starts with the test phase, and the policy must reach and maintain optimality within this test phase. An important constraint in the online setting is that the agent gets to experience each interaction only once. There is no way to revisit past interactions and try a different action in the same context. This is a key distinction with training in a separate offline phase, such as in meta-learning (Al-Shedivat et al., 2018), where the agent can explore the same conditions many times.

Note that the policy that maximizes lifelong return $\pi^*=arg\max_{\pi}J(\pi,\mathbf{z})$ has to be *prescient*, i.e., it needs to have upfront knowledge of the context process $\mathbf{z}$. Since an online agent is causal and has only observed context values up to the current time step t, it can never perform as well as this prescient policy. Therefore, in general online RL agents will have a gap with prescient policies in terms of lifelong return. In certain special cases the online RL can asymptotically reach the prescient policy, e.g., when the context process is Markovian the entire context-driven MDP collapses to a standard MDP with a state $\tilde{s}_t=<s_t,z_t>$. However, we do not intend to limit the context process in any way, and our aim it to minimize the gap between the online and prescient agents for arbitrary context processes.

3 Related Work

Non-stationary RL. Non-stationary RL is a family of sub-problems, such as CF, latent context inference, meta-learning, etc (Khetarpal et al., 2020). In this work we focus on CF, and highlight the differences of CF with other well-known non-stationary RL problems below. Then, we will explore related work for CF in Machine Learning (ML) and RL. We highlight other lines of work in §A in the Appendix.

Latent Context Inference. These works consider a context-driven MDP where the context z_t is unobserved. The goal is to infer an estimated context $\hat{z}_t$ from other signals, such as transition functions, reward functions, etc (Hallak et al., 2015; Zintgraf et al., 2019; Xie et al., 2020; Caccia et al., 2020; Lee et al., 2020; He et al., 2020; Poiani et al., 2021; Chen et al., 2022; Huang et al., 2022; Feng et al., 2022; Ren et al., 2022; Woo et al., 2022; Bing et al., 2022; Luo et al., 2022; Lee et al., 2023). Once inferred, a traditional RL algorithm such as Soft Actor Critic (SAC) learns a policy $\pi(\cdot|s_t,\hat{z}_t)$ from the state and inferred context, and is typically compared to an 'upper-bound policy' that observes the true context $\pi(\cdot|s_t,z_t)$. These works aim to recover the unobserved context, while we focus on CF after observing the true/recovered context. In fact, the 'upper-bound' policies in these works are baselines we compare to in §5. Combining LCPO with this line of work to solve CF in environments with latent context is an interesting future work.

Catastrophic Forgetting. Three general techniques exist for mitigating CF in ML (Parisi et al., 2019); (1) regularizing the optimization to avoid memory loss during sequential training (Kirkpatrick et al., 2017; Zenke et al., 2017; Farajtabar et al., 2019; Lopez-Paz & Ranzato, 2022); (2) training separate parameters per task, and expanding/shrinking parameters as necessary (Rusu et al., 2016; Shmelkov et al., 2017; Li et al., 2019); (3) rehearsal, i.e. retraining on original data or generative batches (Shin et al., 2017; Isele & Cosgun, 2018; Atkinson et al., 2021); or combinations of these techniques (Schwarz et al., 2018; Aljundi et al., 2019).

Regularization techniques such as Elastic Weight Consolidation (EWC) and Orthogonal Gradient Descent (OGD) require task labels. Approximations have been proposed for

problems without task labels (Schwarz et al., 2018) or boundaries (Woo et al., 2022). Kaplanis et al. (2018) use biologically inspired models of brain synapses to regularize networks.

Another class of approaches aim to infer task labels, by learning the transition dynamics of the MDP, and detecting a new environment when a surprising sample is observed with respect to the learned model (Doya et al., 2002; da Silva et al., 2006; Padakandla et al., 2020; Alegre et al., 2021). The inferred labels are often used to train separate policies/models to mitigate CF. These methods are effective when MDP transitions are abrupt and have well-defined boundaries, but are brittle and perform poorly in realistic environments with noisy and hard-to-distinguish non-stationarities (Hamadanian et al., 2022).

For rehearsal, we can use learned models of the MDP to replay past experiences (Xu et al., 2020; Lee et al., 2020; Pong et al., 2020; Huang et al., 2021; Janner et al., 2021). An alternative type of rehearsal is off-policy training (Haarnoja et al., 2018b; van Hasselt et al., 2016) (e.g., Experience Replay (Mnih et al., 2013)), which can train on stale data, naturally circumvent sequential learning and avoid CF. However, off-policy RL is empirically unstable and sensitive to hyperparameters due to bootstrapping and function approximation (Sutton & Barto, 2018), and is often outperformed by on-policy algorithms in online settings (Duan et al., 2016; Gu et al., 2016; Haarnoja et al., 2018b). CLEAR (Rolnick et al., 2019) is an off-policy RL algorithm explicitly designed to overcome CF with fast adaptations. Similarly, PT-DQN (Anand & Precup, 2023) learns a permanent Q-network to remember past tasks while learning a transient Q-network for fast adaptation.

Constrained Optimization. LCPO's constrained optimization formulation is structurally similar to Trust Region Policy Optimization (TRPO) (Schulman et al., 2015), despite our problem being different than TRPO.

4 Locally-Constrained Policy Optimization

Our goal is to learn a policy $\pi(\cdot,\cdot)$ that takes action $a_t \sim \pi(s_t, z_t)$, in a context-driven MDP characterized by an exogenous non-stationary context process.

4.1 Illustrative Example

Consider a simple environment with a discrete context. In this grid-world problem depicted in Figure 1, the agent can move in 4 directions in a 2D grid, and incurs a base negative reward of -1 per step until it reaches the terminal exit state (no penalty in the last step) or fails to within 20 steps. The grid can be in two modes; 1) 'No Trap' mode, where the center cell is empty, and 2) 'Trap Active' mode, where walking into the center cell incurs a reward of -10. When in 'No Trap' mode, the optimal path passes through the center cell, and the best episodic return is -3. In the 'Trap Active' mode, the center cell's penalty forces the optimal path to go left at the blue cell for an optimal episodic return of -5. This environment mode is our discrete context and the source of non-stationarity in this simple example. The agent observes its current location and the context, i.e. whether the trap is on the grid ($z_t = 1$) or not ($z_t = 0$) in every episode (beginning from the start square).

Advantage Actor Critic (A2C). We use the A2C algorithm to train a policy for this environment, while its context changes every so often. Figure 1c depicts the episodic return across time and Figures 1d and 1e depict the total variation distance between the optimal and learned policy when the policy input is 'No Trap' mode (Figure 1d) or 'Trap Active' mode (Figure 1e). This distance represents how close the learned policy is to the optimal in either context. The agent initially attains optimality for the 'No Trap' mode, but once the context changes at epoch 4K it immediately forgets it. Note that during epochs 4K-16K, the A2C agent is only trained on samples from the 'Trap Active' mode, and it's output for the 'No Trap' mode is drifting. When the context changes back to the 'No Trap' mode at epoch 16K, the agent behaves sub-optimally (epochs 16K-18K) before relearning. Figure 1e shows that A2C also forgets the optimal 'Trap Active' policy during the final 4K epochs.

Key Insight. Since the policy observes the current context z_t, it should be able to distinguish between different environment modes. Therefore, if the agent could surgically modify its policy on the current state-context pairs $\pi(\cdot|s_t, z_t)$ and leave outputs for other state-context pairs $\pi(\cdot|s_t, z' \neq z_t)$ unchanged, it would eventually learn a good policy for all contexts. In fact, tabular RL achieves this trivially in this finite discrete state-context space. To illustrate, we apply a

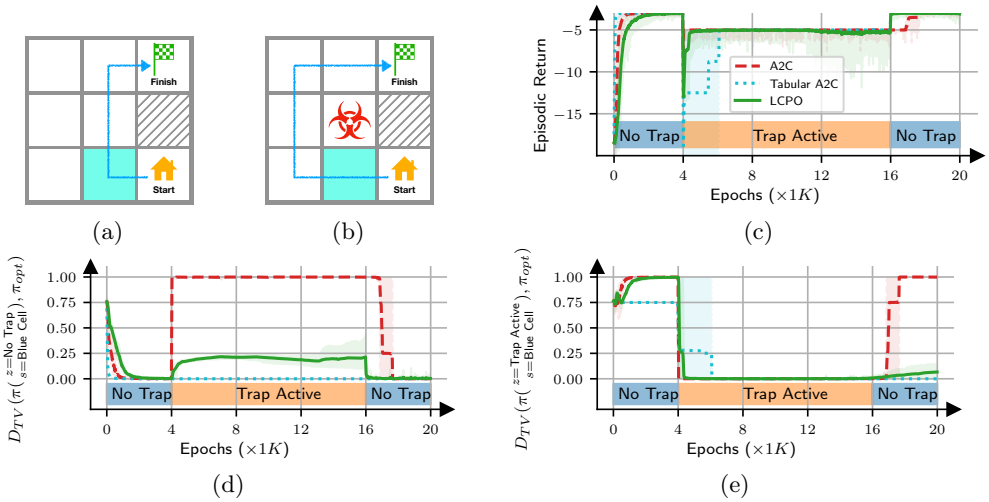


Figure 1: A 3x3 grid-world problem with two modes and the optimal path visualized in blue. **(a)** In the 'No Trap' mode, the center square is safe to pass through. **(b)** In the 'Trap' mode, the agent must avoid the trap with a longer path. **(c)** Episodic return across time in the grid environment. **(d and e)** Total variation distance between learned and optimal policy outputs for the **(d)** 'No Trap' mode, and the **(e)** 'Trap Active' mode at the blue cell (lower is better). Tabular A2C and LCPO remember the optimal decision for either context during shaded regions and instantly attain optimal returns when the environment switches.

tabular version of A2C: i.e., the policy and value networks are replaced with tables with separate rows for each state-context pair (18 total rows). Figures 1d and 1e demonstrate that the tabular RL policy for each context remains unchanged when it does not actively interact with that context. This is because when an experience is used to update the table, it only updates the row pertaining to its own state and context, and does not change rows belonging to other contexts. Under sufficient conditions, tabular RL can provably converge to the optimal policy for such environments. Due to space constraints, we state the theorem and its proof in §B in the Appendix.

Can we achieve a similar behavior with neural network function approximators? In general, updating a neural network (say, a policy network) for certain state-context pairs will change the output for all state-context pairs, leading to CF. But if we could somehow "anchor" the output of the neural network on distinct prior state-context pairs (analogous to the cells in the tabular setting) while we update the relevant state-context pairs, then the neural network would not "forget".

LCPO. Achieving the aforementioned anchoring does not require task labels. We only need to know if two contexts z_i and z_j are *different*. In particular, let the batch of recent environment interactions $< s_t, z_t, a_t, r_t >$ be B_r and let all previous interactions (from possibly different contexts) be B_a. Suppose we have a difference detector $W(B_a, B_r)$ that can be used to sample experiences from B_a that are not from the same distribution as the samples in the recent batch B_r, i.e., the difference detector provides out-of-distribution (OOD) samples with respect to B_r. Then, when optimizing the policy for the current batch B_r, we can constrain the policy's output on experiences sampled via $W(B_a, B_r)$ to not change (see §4.2 for details). We name this approach Locally Constrained Policy Optimization (LCPO). The result for LCPO is presented in Figures 1d and 1e. While it does not retain its policy as perfectly as tabular A2C, it does sufficiently well to recover near instantaneously upon the second switch at epoch 16K.

Change-Point Detection (CPD) vs. OOD Detection. CPD (and task labeling in general) requires stronger assumptions than OOD detection. The context process has to be piece-wise stationary to infer task labels and context changes must happen infrequently to be detectable. Furthermore, online CPD is sensitive to outliers. In contrast, OOD is akin to defining a distance metric on the context process and can be well-defined on any context process. Consider the context process shown in Figure 2. We run this context process through a CPD algorithm (Alegre et al., 2021) for two different sensitivity factors σ_{mbcd}, and represent each detected change-point with a red vertical line. A slight increase in sensitivity leads to 34 detected change-points, and these change-points are also not reasonable. There is no obvious way to assign task labels for this smooth process and there aren't clearly separated

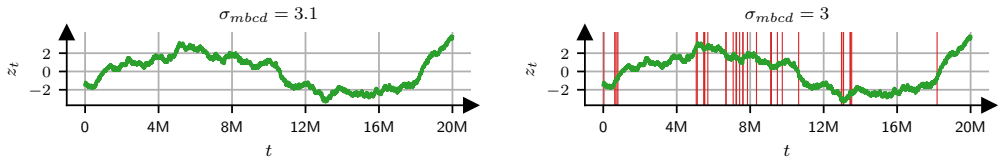


Figure 2: A sample context process z_t, and detected change-points at two thresholds. Teasing meaningful task boundaries is difficult for this process, but defining an OOD metric is intuitive.

segments that can be defined as tasks. However, an intuitive OOD detection method is testing if the distance between z_i and z_j is larger than some threshold, i.e., $|z_i - z_j| > 1$. Altogether, OOD is considerably easier in practice compared to CPD. Note that although the grid-world example – and discrete context environments in general – is a good fit for CPD, this environment was purposefully simple to explain the insight behind LCPO.

4.2 Methodology

Consider a parameterized policy π_θ with parameters θ. Our task is to choose a direction for changing θ such that it improves the expected return on the most recent batch of experiences B_r, while the policy is 'anchored' on prior samples with sufficiently distinct context distributions, $W(B_a, B_r)$.

Algorithm 1 LCPO Training

```
initialize parameter vectors θ_0, empty buffer B_a
for each iteration do
    B_r ← Sample a mini-batch of new interactions
    S_c ← W(B_a, B_r)
    v ← ∇_θ L_tot(θ; B_r)|_{θ_0}
    if S_c is not empty then
        g(x) := ∇_θ(x^T ∇_θ D_KL(θ_old, θ; S_c)|_{θ_0})|_{θ_0}
        v_c ← conjgrad(v, g(·))
        while θ_old + v_c violates constraints do
            v_c ← v_c/2
        θ_0 ← θ_0 + v_c
    else
        θ_0 ← θ_0 + v
    B_a ← B_a + B_r
```

In supervised learning, this anchoring is straightforward to perform, e.g. by adding a regularization loss that directs the neural network to output the ground truth labels for OOD samples (Caruana, 1997). In the case of an RL policy, however, we do not know the ground truth (optimal actions) for anchoring the policy output. Moreover, using the actions we took in prior contexts as the ground truth is not possible, since the policy may have not converged at those times. Anchoring to those actions may cause the policy to relearn suboptimal actions from an earlier period in training. To avoid these problems, LCPO solves a constrained optimization problem that forces the policy to not change for OOD samples. Formally, we consider the following optimization problem:

$$\begin{aligned} \min_{\theta} \;\; & \mathcal{L}_{tot}(\theta; B_r) \triangleq \mathcal{L}_{PG}(\theta; B_r) + \mathcal{L}_e(\theta; B_r) \\ s.t. \;\; & D_{KL}(\theta_0, \theta; W(B_a, B_r)) \leq c_{anchor} \end{aligned} \tag{1}$$

We use the standard definition of policy gradient loss, that optimizes a policy to maximize returns (Schulman et al., 2018; Mnih et al., 2016; Sutton & Barto, 2018):

$$\mathcal{L}_{PG}(\theta; B_r) = \mathbb{E}_{r_t \sim B_r}\left[\sum_{t=0}^{H} -\gamma^t r_t\right] \tag{2}$$

We use automatic entropy regularization (Haarnoja et al., 2018c), to react to and explore in response to novel contexts. The learnable parameter θ_e is adapted such that the entropy coefficient is e^{θ_e}, and the entropy remains close to a target entropy $\bar{\mathcal{H}}$. This worked well in our experiments but LCPO could use any exploration method designed for non-stationary context-driven environments.

$$\mathcal{L}_e(\theta; B_r) = e^{\theta_e} \mathbb{E}_{s_t, z_t \sim B_r, a_t \sim \pi}[\log \pi(a_t | s_t, z_t)] \tag{3}$$

We use KL-divergence as a measure of policy change, and for simplicity we use $D_{KL}(\theta_0,\theta;W(B_a,B_r))$ as a shorthand for $\mathbb{E}_{s,z\sim W(B_a,B_r)}[D_{KL}(\pi_{\theta_0}(s,z)||\pi_\theta(s,z))]$. Here, θ_0 denotes the current policy parameters, and we are solving the optimization over θ to determine the new policy parameters.

Buffer Management. To avoid storing all interactions in B_a we use reservoir sampling (Vitter, 1985), where we randomly replace old interactions with new ones with probability $\frac{n_b}{n_s}$, where n_b is the buffer size and n_s is the total interactions thus far. Reservoir sampling guarantees that the interactions in the buffer are a uniformly random subset of the full set of interactions. For a pseudo-code see §C.1 in the Appendix.

Difference detector. To realize $W(B_a,B_r)$, we treat it as an OOD detection task. A variety of methods can be used in practice (§5), e.g., we can compute the Mahalanobis distance (Mahalanobis, 2018) — the normalized distance of each experience's context with respect to the average context in B_r — and deem any distance above a certain threshold to be OOD. To avoid a high computational overhead when sampling from $W(B_a,B_r)$, we sample a larger batch from B_a, and keep the state-context pairs that are OOD with respect to B_r. If not enough different samples exist, we do not apply the constraint for that update. For a pseudo-code and further implementation details about the OOD detector, see §C.2 and §C.3.

Solving the constrained optimization. To solve this constrained optimization, we approximate the optimization goal and constraint, and calculate a search direction accordingly (pseudocode in Algorithm 1). Our problem is structurally similar to TRPO (Schulman et al., 2015), though the constraint is quite different. Similar to TRPO, we model the optimization goal with a first-order approximation, i.e. $\mathcal{L}_{tot}(\theta;\cdot)=\mathcal{L}_0+(\theta-\theta_0)^T\nabla_\theta\mathcal{L}_{tot}(\theta;\cdot)|_{\theta_0}$, and the constraint with a second order approximation $D_{KL}(\theta_0,\theta;\cdot)=(\theta-\theta_0)^T\nabla^2_\theta D_{KL}(\theta_0,\theta;\cdot)|_{\theta_0}(\theta-\theta_0)$. The optimization problem can therefore be written as

$$\begin{aligned} &\min_{\theta}\ (\theta-\theta_0)^T v \\ &s.t.\ (\theta-\theta_0)^T A(\theta-\theta_0)\leq c_{anchor} \end{aligned} \tag{4}$$

where $A_{ij}=\frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}D_{KL}(\theta_0,\theta;W(B_a,B_r))$, and $v=\nabla_\theta\mathcal{L}_{tot}(\theta;\cdot)|_{\theta_0}$. This optimization problem can be solved using the conjugate gradient method followed by a line search (Schulman et al., 2015; Achiam et al., 2017).

Bounding policy change. The above formulation does not bound policy change on the current context, which could destabilize learning. We could add a second constraint, i.e. TRPO's constraint, $D_{KL}(\theta_0,\theta;B_r)\leq c_{recent}$ (note that this constraint is different from that in Equation (1), as the samples come from B_r instead of $W(B_a,B_r)$). However, having two second order constraints is computationally expensive. Instead, we guarantee the TRPO constraint in the line search phase (lines 9–10 in Algorithm 1), where we repeatedly decrease the gradient update size until both constraints are met.

5 Evaluation

We evaluate LCPO across six environments: four from Gymnasium Mujoco (Towers et al., 2023), one from Gymnasium Classic Control (Towers et al., 2023), and a straggler mitigation task from computer systems (Mao et al., 2019a; Hamadanian et al., 2022). These environments are subject to synthetic or real context processes that affect their dynamics. Our experiments aim to answer the following questions: **(1)** How does LCPO compare to baselines, and can it perform as well as the pre-trained prescient policies (§5.1)? **(2)** How does the accuracy of the OOD sampler $W(\cdot,\cdot)$ affect LCPO (§5.2)? **(3)** How does the maximum buffer size $n_b=|B_a|$ affect LCPO (§5.3)? We include further ablations of LCPO and baselines in Appendices §E.1, §D.5 and §D.1

Baselines. We consider the following approaches for comparison: **Regularization-based:** (1) Online EWC (Kirkpatrick et al., 2017; Chaudhry et al., 2018; Schwarz et al., 2018), (2) Sliding OGD (Farajtabar et al., 2019; Woo et al., 2022) and (3) Benna-Fusi DQN (BFQDN) (Kaplanis et al., 2018), **Task Inference:** (4) Model-Based Changepoint Detection (MBCD) (Alegre et al., 2021), **Rehearsal:** (5) Model-Based Policy Optimization (MBPO) (Janner et al., 2021), (6) CLEAR (Rolnick et al., 2019), (7) PT-DQN (Anand & Precup, 2023), (8) SAC (Haarnoja et al., 2018b) and (9) Double Deep Q Network (DDQN) (Hasselt et al., 2016) **On-policy**

RL: (10) A2C (Mnih et al., 2016) and (11) TRPO (single-path) (Schulman et al., 2015), both using Generalized Advantage Estimation (GAE) (Schulman et al., 2018), **Prescient RL:** (12) as described in §2, the best of policies trained with A2C (Mnih et al., 2016), TRPO (single-path) (Schulman et al., 2015), DDQN (Hasselt et al., 2016) and SAC (Haarnoja et al., 2018b). For more details about these baselines, refer to §D in the Appendix

Experiment Setup. We use 25 random seeds for gymnasium (5 for slower schemes) and 10 random seeds for the straggler mitigation experiments, and use the same hyperparameters for LCPO in all environments and contexts. Gym environments were modified to accept discrete action space policies, as even prescient policies struggled to learn stable continuous space policies in the presence of contexts (See §F.3). Hyperparameters and neural network structures are noted in Appendices §F.4 and §G.2. These experiments were conducted on a machine with 2 AMD EPYC 7763 CPUs (256 logical cores) and 512 GiB of RAM. With 32 concurrent runs, experiments finished in $\sim$1082 hours. This figure does not include runtime devoted to tuning the baselines.

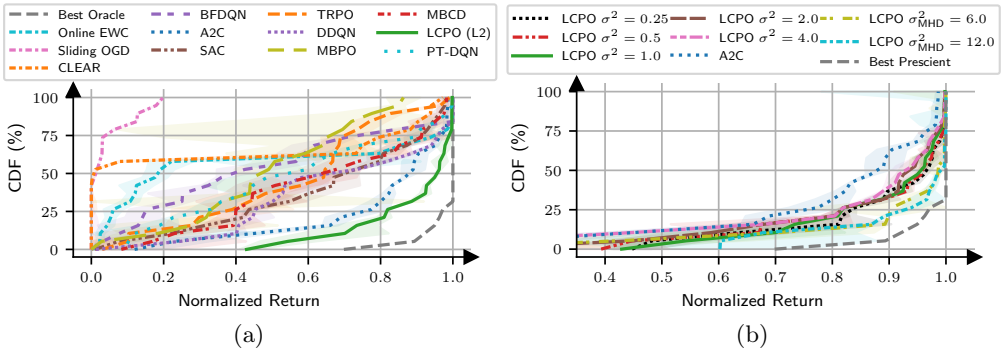
Environment and Contexts. We consider six environments: Modified versions of Pendulum-v1 from the classic control environments, InvertedPendulum-v4, InvertedDoublePendulum-v4, Hopper-v4 and Reacher-v4 from the Mujoco environments (Towers et al., 2023), and a straggler mitigation environment (Hamadanian et al., 2022). In the gym environments, the context is an exogenous "wind" process that creates external force on joints and affects movements. We append the external wind vectors from the last 3 time-steps to the observation, since the agent cannot observe the external context that is going to be applied in the next step, and a history of prior steps helps with the prediction. We create 4 synthetic context sequences with the Ornstein–Uhlenbeck process (Uhlenbeck & Ornstein, 1930), piece-wise Gaussian models, or hand-crafted signals with additive noise. These context processes cover smooth, sudden, stochastic, and predictable transitions at short horizons. All context traces are visualized in Figure 7 in the Appendix. Context traces 1 and 2 are 20 million, and context traces 3 and 4 are 8 million steps long. All baselines were allowed a 'warm-up' period of 6 million time steps, and episodes were truncated at 200 steps. For the straggler mitigation environments, we use workloads provided by the authors in (Hamadanian et al., 2022), that are from a production web framework cluster at AnonCo, collected from a single day in February 2018. These workloads are visualized in Figure 8b in the appendix.

OOD detection. We set the buffer size n_b to 1% of all samples, which is $n_b \leq 200K$. To sample OOD state-context pairs $W(B_a, B_r)$, we use distance metrics and thresholds. For gym environments where the context is a wind process, we use L2 distance, i.e. if $\overline{w_r} = \mathbb{E}_{w \sim B_r}[w]$ is the average wind vector observed in the recent batch B_r, we sample a minibatch of states in B_a where $W(B_a, B_r) = \{w_i | \forall w_i \in B_a : \|w_i - \overline{w_r}\|_2 > \sigma\}$. There exist domain-specific models for workload distributions in the straggler mitigation environment, but we used Mahalanobis distance as it is a well-accepted and general approach for outlier detection in prior work (Lee et al., 2018; Podolskiy et al., 2021). Concretely, we fit a Gaussian multivariate model to the recent batch B_r, and report a minibatch of states in B_a with a Mahalanobis distance further than σ from this distribution (see §C.2 in the Appendix for more details).

5.1 Results

To evaluate across different gymnasium environments and traces, we score agents with *Normalized Return*, i.e., for each environment and context process we report a scaled score function where 0 and 1 are the minimum and maximum lifelong return across all agents. We prefer agents with higher scores over different environments and traces. Figure 3a provides a summary of all gymnasium experiments (full details in Table 5 in the Appendix). LCPO maintains a lead over baselines, is close to the best-performing prescient policy, all while learning online and sequentially. We present a detailed analysis of baselines' performance in §D, and we summarize these findings below. We also report the wallclock time for each scheme in §F.2 in the appendix; LCPO is $\sim$1.5$\times$ as demanding as A2C.

Online EWC and Sliding OGD employ heuristics to circumvent the necessity of task labels in the original techniques. Conceptually, they implicitly assume past episodes are separate "tasks". Empirically these heuristics are not successful at solving CF. As for BFDQN, Kaplanis et al. (2018) note that while the architecture was successful in simple environments, it failed with more complex and challenging ones. In our experience, this architecture did not provide any benefits compared to vanilla DDQN.



(a) (b)

Figure 3: CDF of normalized lifelong returns, where 0/1 denote the lowest/highest returns among agents. Shaded regions denote 95% confidence intervals. **(a)** LCPO outperforms all online agents, and remains the closest to prescient policies. **(b)** LCPO is affected by the OOD threshold σ, but still maintains its lead over baselines.

MBCD struggles to tease out meaningful task boundaries. In some experiments, it launches anywhere between 3 to 7 policies just by changing the random seed. This observation is in line with MBCD's sensitivity in §4.1, and prior work (Hamadanian et al., 2022).

MBPO performed poorly, even though we verified that its learned model of the environment is highly accurate. CLEAR (Rolnick et al., 2019) and PT-DQN (Anand & Precup, 2023) are highly hyperparameter sensitive due to how they address catastrophic forgetting. While we tuned both extensively for the Pendulum-v1 environment, as we did for all baselines, they fail catastrophically in other environments. SAC and Deep Q Network (DQN) struggle to outperform A2C in the online case. This falls in line with prior observations (Hamadanian et al., 2022) and can be attributed to the instability and hyperparameter sensitivity of off-policy RL (Duan et al., 2016; Gu et al., 2016; Haarnoja et al., 2018b) and the quick adaptation that a fully online algorithm such as A2C provides (Sutton et al., 2007). In fact, despite not having been designed for non-stationary RL, A2C is the most successful baseline.

Table 1: Tail latency (negative reward) and 95th percentile confidence ranges for different algorithms and contexts in the straggler mitigation environment.

	Online Learning										
	LCPO Agg	LCPO Med	LCPO Cons	MBCD	MBPO	Online EWC	A2C	TRPO	DDQN	SAC	Best Prescient
Workload 1	1070 ±10	1076 ±16	1048 ±7	1701 ±112	2531 ±197	2711 ±232	1716 ±710	3154 ±464	1701 ±47	1854 ±245	984 (TRPO)
Workload 2	589 ±43	617 ±62	586 ±27	678 ±38	891 ±54	724 ±22	604 ±109	864 ±105	633 ±7	644 ±27	509 (A2C)

For the straggler mitigation environment, Table 1 presents the latency metric (negative reward) over two workloads. Recall that this environment uses real-world traces from a production cloud network. The overall trends are similar to the gymnasium experiments, with LCPO outperforming all other baselines. This table includes three variants of LCPO, that will be discussed further in §5.2.

5.2 Sensitivity to OOD metric

LCPO applies a constraint to OOD state-context pairs, as dictated by the OOD sampler $W(B_a, B_r)$. We vary the OOD threshold σ—which the OOD method uses in sampling—and monitor the normalized return for the gym environments in Figure 3b and the straggler mitigation environments in Table 1. In the gym environments, a lower value for σ yields tighter margin of difference before a sample is deemed OOD. LCPO is affected by σ, with the smallest threshold $\sigma^2 = 0.25$ performing the best. However, LCPO still maintains a lead over the A2C baseline across σ variations. We also experiment with a handicapped OOD metric that observes a state-context vector $x_t = < s_t, z_t >$ *without the ability to separate state and context.* We use the Mahalanobis distance OOD metric (Mahalanobis, 2018) at several thresholds σ^2_{MHD} for this experiment. Despite the handicap, the LCPO +Mahalanobis surpasses the LCPO +L2 agent that we have used in this evaluation. This is not surprising, as the L2 distance is less robust than

Mahalanobis distance, but easier to interpret. In the straggler mitigation environment LCPO Agg, LCPO Med and LCPO Cons use $\sigma = 5$, 6 and 7, and a higher value for σ yields more conservative OOD samples (i.e., fewer samples are detected as OOD). The difference between these three is significant: The model in LCPO Agg allows for 26.7× more samples to be considered OOD compared to LCPO Cons. Table 1 provides the normalized return for LCPO with varying thresholds, along with baselines. Notably, all variations of LCPO achieve similar results.

5.3 Sensitivity to Buffer Size

LCPO uses Reservoir sampling (Vitter, 1985) to maintain a limited number of samples n_b. We evaluate how sensitive LCPO is to the buffer size in Figure 4 (full results in Table 7 in the Appendix). The full experiment has 8–20M samples. LCPO maintains its high performance, even with as little as $n_b = 500$ samples, but drops below this point (statistically significant in over one thirds of experiments). This is not surprising, as the context traces do not change drastically at short intervals, and even 500 randomly sampled points from the trace should be enough to have a representation over all of the trace. However, with more complicated and high-dimensional contexts, a higher buffer size would likely be necessary.

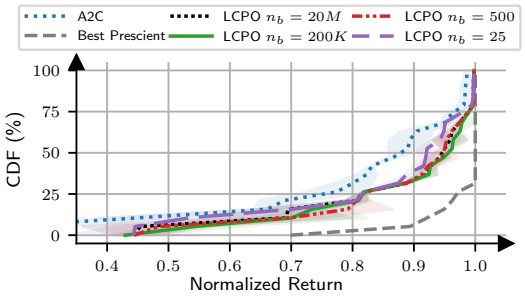


Figure 4: CDF of normalized returns of LCPO in gym environments with various buffer sizes. Shaded regions denote 95% confidence intervals. LCPO loses performance with $n_b < 500$.

6 Discussion and Limitations

Network Capacity. In general, online learning methods with bounded parameter counts will reach the function approximator's (neural network's) maximum representational capacity. LCPO is not immune from this, as we do not add parameters with more context traces. However, neither are prescient agents. To isolate the effect of this capacity and CF, we compare against prescient agents, rather than single agents trained on individual tasks or context traces (He et al., 2020). This ensures a fair evaluation that does not penalize online learning for reaching the capacity ceiling. If the maximum capacity has been reached, it may be beneficial to remove significantly old samples from B_a to allow LCPO to forget such contexts, thereby favoring flexibility instead of stability.

Exploration. LCPO focuses on mitigating catastrophic forgetting in non-stationary RL. An orthogonal challenge in this setting is efficient exploration, i.e. to explore when the context distribution has changed but only once per new context. Our experiments used automatic entropy tuning for exploration (Haarnoja et al., 2018b), which, while empirically effective, was not designed for non-stationary problems. LCPO may benefit from a better exploration methodology such as curiosity-driven exploration (Pathak et al., 2017).

Efficient Buffer Management. We used Reservoir Sampling (Vitter, 1985), which maintains a uniformly random buffer of all observed state-context samples so far. Future work could explore strategies that selectively store or drop samples based on their context, e.g., to maximize sample diversity.

7 Conclusion

We proposed and evaluated LCPO, a simple approach for online learning in non-stationary context-driven environments. LCPO requires two conditions: (1) the non-stationarity must be induced by an exogenous observed context process; and (2) a similarity metric is required that can inform us if two contexts come from noticeably different distributions (OOD detection). This is less restrictive than prior approaches that require either explicit or inferred task labels. Our experiments showed that LCPO outperforms baselines on several environments with real and synthetic context processes.

Reproducibility Statement

For Theorem B.2, we include the proof and assumptions in §B. We include detailed accounts of environments, context traces, baselines, hyperparameters, software and hardware in §5 in the main text and §F and §G in the Appendix. We include implementation details and pseudo algorithms in §4.2 and §C in the Appendix. A link to an anonymized source code is also provided in §1.

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization, 2017.

Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments, 2018.

Lucas N Alegre, Ana LC Bazzan, and Bruno C da Silva. Minimum-delay adaptation in non-stationary reinforcement learning via online high-confidence change-point detection. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 97–105, 2021.

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning, 2019.

Nishanth Anand and Doina Precup. Prediction and control in continual reinforcement learning, 2023. URL `https://arxiv.org/abs/2312.11669`.

Craig Atkinson, Brendan McCane, Lech Szymanski, and Anthony Robins. Pseudo-rehearsal: Achieving deep reinforcement learning without catastrophic forgetting. *Neurocomputing*, 428:291–307, Mar 2021. ISSN 0925-2312. doi: 10.1016/j.neucom.2020.11.050. URL `http://dx.doi.org/10.1016/j.neucom.2020.11.050`.

Emmanuel Bengio, Joelle Pineau, and Doina Precup. Interference and generalization in temporal difference learning, 2020. URL `https://arxiv.org/abs/2003.06350`.

Zhenshan Bing, Lukas Knak, Fabrice Oliver Robin, Kai Huang, and Alois Knoll. Meta-reinforcement learning in broad and non-parametric environments, 2022.

Massimo Caccia, Pau Rodriguez, Oleksiy Ostapenko, Fabrice Normandin, Min Lin, Lucas Page-Caccia, Issam Hadj Laradji, Irina Rish, Alexandre Lacoste, David Vázquez, et al. Online fast adaptation and knowledge accumulation (osaka): a new approach to continual learning. *Advances in Neural Information Processing Systems*, 33:16532–16545, 2020.

Rich Caruana. Multitask learning. *Machine learning*, 28:41–75, 1997.

Yash Chandak, Georgios Theocharous, Shiv Shankar, Martha White, Sridhar Mahadevan, and Philip S. Thomas. Optimizing for the future in non-stationary mdps, 2020.

Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. *Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence*, pp. 556–572. Springer International Publishing, 2018. ISBN 9783030012526. doi: 10.1007/978-3-030-01252-6_33. URL `http://dx.doi.org/10.1007/978-3-030-01252-6_33`.

Xiaoyu Chen, Xiangming Zhu, Yufeng Zheng, Pushi Zhang, Li Zhao, Wenxue Cheng, Peng Cheng, Yongqiang Xiong, Tao Qin, Jianyu Chen, et al. An adaptive deep rl method for non-stationary environments with piecewise stable context. *Advances in Neural Information Processing Systems*, 35:35449–35461, 2022.

Bruno C. da Silva, Eduardo W. Basso, Ana L. C. Bazzan, and Paulo M. Engel. Dealing with non-stationary environments using context detection. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pp. 217–224, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933832. doi: 10.1145/1143844.1143872. URL `https://doi.org/10.1145/1143844.1143872`.

Thomas G. Dietterich, George Trimponias, and Zhitang Chen. Discovering and removing exogenous state variables and rewards for reinforcement learning. In *International Conference on Machine Learning*, 2018. URL https://api.semanticscholar.org/CorpusID:46938244.

Kenji Doya, Kazuyuki Samejima, Ken-ichi Katagiri, and Mitsuo Kawato. Multiple model-based reinforcement learning. *Neural computation*, 14(6):1347–1369, 2002.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pp. 1329–1338. PMLR, 2016.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures, 2018.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning, 2019. URL https://arxiv.org/abs/1910.07104.

Fan Feng, Biwei Huang, Kun Zhang, and Sara Magliacane. Factored adaptation for non-stationary reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 31957–31971, 2022.

Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning, 2019.

Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E Turner, and Sergey Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. *arXiv preprint arXiv:1611.02247*, 2016.

Zhouyou Gu, Changyang She, Wibowo Hardjawana, Simon Lumb, David McKechnie, Todd Essery, and Branka Vucetic. Knowledge-assisted deep reinforcement learning in 5g scheduler design: From theoretical framework to implementation. *IEEE Journal on Selected Areas in Communications*, 39(7):2014–2028, 2021.

Tuomas Haarnoja, Vitchyr Pong, Aurick Zhou, Murtaza Dalal, Pieter Abbeel, and Sergey Levine. Composable deep reinforcement learning for robotic manipulation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6244–6251, 2018a. doi: 10.1109/ICRA.2018.8460756.

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

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications, 2018c. URL https://arxiv.org/abs/1812.05905.

Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes, 2015. URL https://arxiv.org/abs/1502.02259.

Pouya Hamadanian, Malte Schwarzkopf, Siddartha Sen, and Mohammad Alizadeh. Reinforcement learning in time-varying systems: an empirical study, 2022. URL https://arxiv.org/abs/2201.05560.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pp. 2094–2100. AAAI Press, 2016.

Ammar Haydari and Yasin Yılmaz. Deep reinforcement learning for intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(1):11–32, 2022. doi: 10.1109/TITS.2020.3008612.

Xu He, Jakub Sygnowski, Alexandre Galashov, Andrei Alex Rusu, Yee Whye Teh, and Razvan Pascanu. Task agnostic continual learning via meta learning. In *4th Lifelong Machine Learning Workshop at ICML 2020*, 2020. URL https://openreview.net/forum?id=AeIzVxdJgeb.

Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin Riedmiller, and David Silver. Emergence of locomotion behaviours in rich environments, 2017a.

Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin Riedmiller, and David Silver. Emergence of locomotion behaviours in rich environments, 2017b. URL https://arxiv.org/abs/1707.02286.

Biwei Huang, Fan Feng, Chaochao Lu, Sara Magliacane, and Kun Zhang. Adarl: What, where, and how to adapt in transfer reinforcement learning, 2022.

Yizhou Huang, Kevin Xie, Homanga Bharadhwaj, and Florian Shkurti. Continual model-based reinforcement learning with hypernetworks, 2021.

David Isele and Akansel Cosgun. Selective experience replay for lifelong learning, 2018.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization, 2021.

Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Continual reinforcement learning with complex synapses, 2018.

Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *arXiv preprint arXiv:2012.13490*, 2020.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

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.

Qingfeng Lan and A. Rupam Mahmood. Elephant neural networks: Born to be a continual learner, 2023. URL https://arxiv.org/abs/2310.01365.

Qingfeng Lan, Yangchen Pan, Jun Luo, and A. Rupam Mahmood. Memory-efficient reinforcement learning with value-based knowledge consolidation, 2023. URL https://arxiv.org/abs/2205.10868.

Hyunin Lee, Yuhao Ding, Jongmin Lee, Ming Jin, Javad Lavaei, and Somayeh Sojoudi. Tempo adaptation in non-stationary reinforcement learning, 2023.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks, 2018. URL https://arxiv.org/abs/1807.03888.

Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. Context-aware dynamics model for generalization in model-based reinforcement learning, 2020.

Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting, 2019.

Vincent Liu, Raksha Kumaraswamy, Lei Le, and Martha White. The utility of sparse representations for control in reinforcement learning, 2018. URL https://arxiv.org/abs/1811.06626.

Vincent Liu, Han Wang, Ruo Yu Tao, Khurram Javed, Adam White, and Martha White. Measuring and mitigating interference in reinforcement learning, 2023. URL https://arxiv.org/abs/2307.04887.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning, 2022.

Fan-Ming Luo, Shengyi Jiang, Yang Yu, ZongZhang Zhang, and Yi-Feng Zhang. Adapt to environment sudden changes by learning a context sensitive policy. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(7):7637–7646, Jun. 2022. doi: 10.1609/aaai.v36i7.20730. URL https://ojs.aaai.org/index.php/AAAI/article/view/20730.

Prasanta Chandra Mahalanobis. On the generalized distance in statistics. *Sankhyā: The Indian Journal of Statistics, Series A (2008-)*, 80:S1–S7, 2018.

Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pp. 197–210, 2017.

Hongzi Mao, Malte Schwarzkopf, Shaileshh Bojja Venkatakrishnan, Zili Meng, and Mohammad Alizadeh. Learning scheduling algorithms for data processing clusters, 2018. URL https://arxiv.org/abs/1810.01963.

Hongzi Mao, Malte Schwarzkopf, Hao He, and Mohammad Alizadeh. Towards safe online reinforcement learning in computer systems. 2019a.

Hongzi Mao, Malte Schwarzkopf, Shaileshh Bojja Venkatakrishnan, Zili Meng, and Mohammad Alizadeh. Learning scheduling algorithms for data processing clusters. In *Proceedings of the ACM Special Interest Group on Data Communication*, SIGCOMM '19, pp. 270–288, New York, NY, USA, 2019b. Association for Computing Machinery. ISBN 9781450359566. doi: 10.1145/3341302.3342080. URL https://doi.org/10.1145/3341302.3342080.

Hongzi Mao, Shannon Chen, Drew Dimmery, Shaun Singh, Drew Blaisdell, Yuandong Tian, Mohammad Alizadeh, and Eytan Bakshy. Real-world video adaptation with reinforcement learning, 2020.

Ryan Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil, and Nesime Tatbul. Neo: A learned query optimizer. *arXiv preprint arXiv:1904.03711*, 2019.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165, 1989. ISSN 0079-7421. doi: https://doi.org/10.1016/S0079-7421(08)60536-8. URL https://www.sciencedirect.com/science/article/pii/S0079742108605368.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016.

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, 2019a.

Anusha Nagabandi, Chelsea Finn, and Sergey Levine. Deep online learning via meta-learning: Continual adaptation for model-based rl, 2019b.

Sindhu Padakandla, Prabuchandran K. J., and Shalabh Bhatnagar. Reinforcement learning algorithm for non-stationary environments. *Applied Intelligence*, 50(11):3590–3606, jun 2020. doi: 10.1007/s10489-020-01758-5. URL https://doi.org/10.1007%2Fs10489-020-01758-5.

E. S. PAGE. CONTINUOUS INSPECTION SCHEMES. *Biometrika*, 41(1-2):100–115, 06 1954. ISSN 0006-3444. doi: 10.1093/biomet/41.1-2.100. URL https://doi.org/10.1093/biomet/41.1-2.100.

Minting Pan, Xiangming Zhu, Yunbo Wang, and Xiaokang Yang. Iso-dream: Isolating noncontrollable visual dynamics in world models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=6LBfSduVg0N.

Yangchen Pan, Kirby Banman, and Martha White. Fuzzy tiling activations: A simple approach to learning sparse representations online, 2021. URL https://arxiv.org/abs/1911.08068.

German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2019.01.012. URL https://www.sciencedirect.com/science/article/pii/S0893608019300231.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction, 2017. URL https://arxiv.org/abs/1705.05363.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning, 2017. URL https://arxiv.org/abs/1703.02702.

Alexander Podolskiy, Dmitry Lipin, Andrey Bout, Ekaterina Artemova, and Irina Piontkovskaya. Revisiting mahalanobis distance for transformer-based out-of-domain detection, 2021. URL https://arxiv.org/abs/2101.03778.

R Poiani, A Tirinzoni, M Restelli, et al. Meta-reinforcement learning by tracking task non-stationarity. In *IJCAI*, pp. 2899–2905. International Joint Conferences on Artificial Intelligence, 2021.

Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. Temporal difference models: Model-free deep rl for model-based control, 2020.

Antonin Raffin. Rl baselines3 zoo. https://github.com/DLR-RM/rl-baselines3-zoo, 2020.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables, 2019.

Hang Ren, Aivar Sootla, Taher Jafferjee, Junxiao Shen, Jun Wang, and Haitham Bou Ammar. Reinforcement learning in presence of discrete markovian context evolution, 02 2022.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. Experience replay for continual learning, 2019.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization, 2015. URL https://arxiv.org/abs/1502.05477.

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

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018.

Jonathan Schwarz, Jelena Luketina, Wojciech M. Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning, 2018.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay, 2017.

Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. Incremental learning of object detectors without catastrophic forgetting, 2017.

Sean R. Sinclair, Felipe Frujeri, Ching-An Cheng, Luke Marshall, Hugo Barbalho, Jingling Li, Jennifer Neville, Ishai Menache, and Adith Swaminathan. Hindsight learning for mdps with exogenous inputs. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Shagun Sodhani, Amy Zhang, and Joelle Pineau. Multi-task reinforcement learning with context-based representations, 2021.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

Richard S. Sutton, Anna Koop, and David Silver. On the role of tracking in stationary environments. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pp. 871–878, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933. doi: 10.1145/1273496.1273606. URL https://doi.org/10.1145/1273496.1273606.

Yunhao Tang and Shipra Agrawal. Discretizing continuous action space for on-policy optimization, 2019. URL https://arxiv.org/abs/1901.10500.

Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning, 2017. URL https://arxiv.org/abs/1707.04175.

Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023. URL https://zenodo.org/record/8127025.

G. E. Uhlenbeck and L. S. Ornstein. On the theory of the brownian motion. *Phys. Rev.*, 36: 823–841, Sep 1930. doi: 10.1103/PhysRev.36.823. URL https://link.aps.org/doi/10.1103/PhysRev.36.823.

Hado van Hasselt, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. Learning values across many orders of magnitude, 2016.

Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

Chen-Yu Wei and Haipeng Luo. Non-stationary reinforcement learning without prior knowledge: An optimal black-box approach, 2021.

Keith Winstein and Hari Balakrishnan. Tcp ex machina: computer-generated congestion control. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pp. 123–134, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450320566. doi: 10.1145/2486001.2486020. URL https://doi.org/10.1145/2486001.2486020.

Honguk Woo, Gwangpyo Yoo, and Minjong Yoo. Structure learning-based task decomposition for reinforcement learning in non-stationary environments. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8):8657–8665, Jun. 2022. doi: 10.1609/aaai.v36i8.20844. URL https://ojs.aaai.org/index.php/AAAI/article/view/20844.

Cathy Wu, Aboudy Kreidieh, Kanaad Parvate, Eugene Vinitsky, and Alexandre M Bayen. Flow: Architecture and benchmarking for reinforcement learning in traffic control. *arXiv preprint arXiv:1710.05465*, 10, 2017.

Annie Xie, James Harrison, and Chelsea Finn. Deep reinforcement learning amidst lifelong non-stationarity, 2020.

Mengdi Xu, Wenhao Ding, Jiacheng Zhu, Zuxin Liu, Baiming Chen, and Ding Zhao. Task-agnostic online reinforcement learning with an infinite mixture of gaussian processes, 2020.

Zhenghai Xue, Qingpeng Cai, Shuchang Liu, Dong Zheng, Peng Jiang, Kun Gai, and Bo An. State regularized policy optimization on data with dynamics shift, 2024. URL https://arxiv.org/abs/2306.03552.

Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization, 2020.

Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8):364–372, 2019.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence, 2017.

Huanhuan Zhang, Anfu Zhou, Yuhan Hu, Chaoyue Li, Guangping Wang, Xinyu Zhang, Huadong Ma, Leilei Wu, Aiyun Chen, and Changhui Wu. Loki: Improving long tail performance of learning-based real-time video adaptation by fusing rule-based models. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, MobiCom '21, pp. 775–788, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383424. doi: 10.1145/3447993.3483259. URL https://doi.org/10.1145/3447993.3483259.

Jinpeng Zhang, Yufeng Zheng, Chuheng Zhang, Li Zhao, Lei Song, Yuan Zhou, and Jiang Bian. Robust situational reinforcement learning in face of context disturbances. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, September 2023. URL https://www.microsoft.com/en-us/research/publication/robust-situational-reinforcement-learning-in-face-of-context-disturbances/.

Henry Zhu, Justin Yu, Abhishek Gupta, Dhruv Shah, Kristian Hartikainen, Avi Singh, Vikash Kumar, and Sergey Levine. The ingredients of real-world robotic reinforcement learning, 2020. URL https://arxiv.org/abs/2004.12570.

Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In *International Conference on Machine Learning*, pp. 7693–7702. PMLR, 2019.

Appendix A Related Work: Continued

Meta-learning. Here, in the general case, the environment can switch between a set of possible MDPs without an explicit signal. At the 'meta-train' phase, the policy is allowed to train on a part or all of the MDPs. At the 'meta-test' phase, the policy must make decisions in a continual RL setup where the MDP may abruptly change, and it has to adapt to the current MDP with few-shot adaptation (Al-Shedivat et al., 2018; Nagabandi et al., 2019b;a) or context inference (Rakelly et al., 2019). In our problem setup, we assume no access to the environment beforehand.

Multi-task RL. In this problem, there are several (e.g., 10 or 50) different tasks with different MDPs. The goal of this line of work is to learn a shared policy for all tasks that approaches the performance of learning separate policies per task (Yang et al., 2020; Sodhani et al., 2021). These tasks may come with contextual information about the task that can be used in the policy (Sodhani et al., 2021). This problem is not continual RL, does not experience CF, and the learner is allowed to explore all tasks at the same time. Often the goal is postitive transfer learning, i.e., learning faster on all tasks in parallel than learning tasks separately (Teh et al., 2017).

Another type of work focuses on speeding up the learning process for a set of new tasks by pre-training on a set of old tasks (Xue et al., 2024). This line of work bears similarities to multi-task RL and meta-learning.

Assuming there are no explicit signals for environment contexts, Wei & Luo (2021) provide a regret-optimal black-box RL algorithm.

Interfernce in stationary RL. Investigating interference in vanilla RL (non-stationarity in sample distribution) is an adjacent and interesting line of work (Bengio et al., 2020; Pan et al., 2021; Liu et al., 2023; 2018). The type of non-stationarity discussed in these works is different than what we study. Here, non-stationarity refers to the moving target of bootstrap losses, such as Q-learning, due to shifting policies that change future data distributions. Non-stationarity in our problem setup means the MDP itself is shifting, irrespective of the changes the policy makes. The second type of non-stationarity cannot be resolved even with "perfect" RL algorithms that deal with interference in stationary MDPs.

However, there may be ideas that are transferable. For example, this line of work suggests that techniques such as GAE (Schulman et al., 2018) and target networks reduce interference (Bengio et al., 2020; Liu et al., 2023). Another interesting avenue is experimentation with sparse learning (Pan et al., 2021; Liu et al., 2018).

Buffer limitation interference. A line of work, related to interference, deals with a type of non-stationarity induced by having small buffers (e.g., 32 samples, compared to the typical thousands to millions) in off-policy algorithms. These works aim to mimic an off-policy agent with unbounded buffers, and do not focus on context-driven non-stationarity. They learn policies that perform closely to unbounded agents via techniques such as following old target Q networks (Lan et al., 2023) or utilizing sparse-gradient activation functions (Lan & Mahmood, 2023). Note that we do compare to DDQN and SAC with unbounded buffers.

Appendix B Proof of Optimality in Tabular Context-driven RL

Below, we show how the non-stationary environment in §2 can be learned with vanilla RL algorithms, if the state, action and context spaces are finite, context switches occur at episode boundaries. First, we prove Lemma §B.1, and then we prove the main theorem Theorem B.2.

Lemma B.1. *Given a monotonically decreasing sequence $\{\alpha_i\}_{i=1}^{\infty}$ that satisfies the following conditions:*

$$\sum_{i=1}^{\infty}\alpha_i=\infty \qquad \sum_{i=1}^{\infty}\alpha_i^2<\infty \tag{5}$$

Consider any subsequence $\{\beta_j\}_{j=1}^{\infty}$, where for any $1\leq j$, there exists $N\times(j-1)<i\leq N\times j$ such that $\beta_j=\alpha_i$.

Prove that:

$$\sum_{j=1}^{\infty}\beta_j=\infty \qquad \sum_{j=1}^{\infty}\beta_j^2<\infty \tag{6}$$

Proof. First, note that since $\{\alpha_i\}_{i=1}^{\infty}$ is monotonically decreasing, we have:

$$\alpha_{N\times i}\leq\beta_i\leq\alpha_{N\times(i-1)+1} \tag{7}$$

For the first equality, we have for all $k>0$:

$$\begin{aligned}\sum_{i=1}^{\infty}\beta_i &\geq \sum_{i=1}^{\infty}\alpha_{N\times i}\\ &\geq \sum_{i=1}^{\infty}\alpha_{N\times i+k}\end{aligned} \tag{8}$$

Thus, we have:

$$\begin{aligned}N\times\sum_{i=1}^{\infty}\beta_i &= \sum_{k=1}^{N}\sum_{i=1}^{\infty}\beta_i\\ &\geq \sum_{k=1}^{N}\sum_{i=1}^{\infty}\alpha_{N\times i+k}\\ &= \sum_{i=N+1}^{\infty}\alpha_i\end{aligned} \tag{9}$$

Therefore:

$$\sum_{i=1}^{\infty}\beta_i\geq\frac{1}{N}\sum_{i=1}^{\infty}\alpha_i-\frac{1}{N}\sum_{i=1}^{N}\alpha_i=\infty \tag{10}$$

The second bound is trivially proven:

$$\begin{aligned}\sum_{i=1}^{\infty}\beta_i^2 &\leq \sum_{i=1}^{\infty}\alpha_{N\times(i-1)+1}^2\\ &\leq \sum_{i=1}^{\infty}\alpha_i^2\\ &<\infty\end{aligned} \tag{11}$$

□

Theorem B.2. *Consider a context-driven MDP as defined in §2. Under the following set of assumptions, prove that the Q-learning algorithm (Watkins & Dayan, 1992) converges to the optimal policy:*

1. *Rewards are bounded bounded rewards $|r_t|\geq R$.*
2. *We have a a monotonically decreasing sequence of learning rates $\{\alpha_i\}_{i=1}^{\infty}$ where $0\geq\alpha_i<1$, and*

$$\sum_{i=1}^{\infty}\alpha_i=\infty \qquad \sum_{i=1}^{\infty}\alpha_i^2<\infty \tag{12}$$

3. *The state, action and context spaces are finite $|\mathcal{S}|,|\mathcal{A}|,|\mathcal{Z}|<\infty$.*
4. *There exists N, such that for any context $z\in\mathcal{Z}$, z occurs at least once in any consecutive subsequence of the context trace of size N.*
5. *The context only changes at episode terminations.*

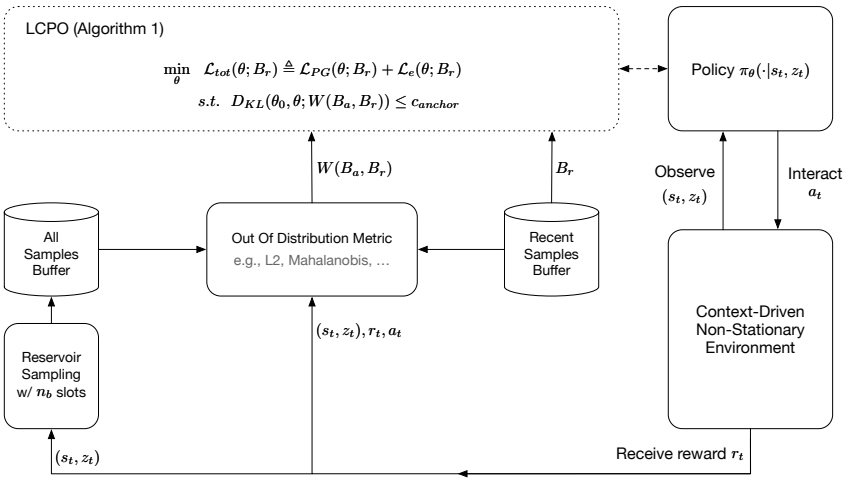


Figure 5: Architecture of LCPO.

Proof. As contexts only switch at episode boundaries, the Q-learning updates for each context $z \in \mathcal{Z}$ are isolated from Q-values of other contexts. Thus, for any context $z \in \mathcal{Z}$, we have a separate Q-learning session. If I_z denotes the set of episode indices where the context trace is equal to z, If we prove

$$\sum_{i \in I_z} \alpha_i = \infty \qquad \sum_{i \in I_z} \alpha_i^2 < \infty \tag{13}$$

we can use the original Q-learning proof of convergence (Watkins & Dayan, 1992).

To prove this, note that from assumption 5 we can surmise that that for any $1 \leq j$, there exists $N \times (j-1) < i \leq N \times j$ such that $i \in I_z$. Therefore based on Lemma §B.1 the subsequence of learning rates satisfy the Watkins Q-learning condition. □

Appendix C LCPO Implementation

Here, we will discuss the implementation details of LCPO. Figure 5 depicts the overall architecture of LCPO.

C.1 Reservoir Sampling

As discussed in §4.2, LCPO needs to maintain a buffer B_a of all samples observed so far. However, this buffer will grow with time and incur extensive memory costs. To limit the buffer size while maintaining a distribution of all samples observed so far, we utilize Reservoir Sampling (Vitter, 1985).

The pseudo-code for the implementation can be found in Algorithm 2. Reservoir sampling operates by maintaining a bounded list of samples B_a. While the number of samples in the list B_a has not reached max capacity n_b, all samples are admitted. Once the buffer is full, a random index is sampled in the range of 0 to $n_s - 1$, where n_s is the number of samples observed so far. If the index is smaller than n_b, the element at index i is replaced with the new sample. If it is larger, the sample is thrown away. This strategy bounds the size of B_a, but maintains a uniform distribution of samples from the true list of all samples observed so far.

C.2 OOD Function

LCPO requires a definition for OOD samples, i.e., samples that come from contexts far away from recent samples B_r. Intuitively, the optimal policy conditioned on this OOD context should be significantly different from the policy optimal conditioned on the current context.

Such metrics can be based on domain insight, where an expert who is familiar with how the context value changes the MDP would define a function to detect OOD samples. Alternatively,

Algorithm 2 Reservoir Sampling

Input: max capacity n_b
$B_a \leftarrow \{\}$ Initialize empty buffer
$n_s \leftarrow 0$ Initialize sample count
for each new sample x **do**
  $n_s \leftarrow n_s + 1$ Increment the sample count
  if $n_s > n_b$ **then**
    $i \sim Unif(0, n_s - 1)$ Sample a random index smaller than n_s
    if $i < n_b$ **then**
      Replace element at index i in B_a with x
    else
      Throw new sample x away
  else
    $B_a \leftarrow B_a + \{x\}$ Append x to B_a

it may be generic OOD criteria commonly used in the literature, such as L2 thresholding, Mahalanobis distance, etc. In this case, the OOD function needs to be tuned to be meaningful, which is a common challenge in OOD detection work. An interesting line of future work is to utilize MDP transitions in the warm-up period for tuning the threshold online.

Concretely, LCPO requires an OOD function $\omega(z, B_r)$ that denotes whether z is OOD with respect to B_r or not. In the L2 thresholding used for the gym environments subject to the wind context in §5, we calculate an average over B_r, i.e., $\mu_w = \mathbb{E}_{w \sim B_r}[w]$, and define $\omega(w, B_r) := \|w - \mu_w\|_2 > \sigma$ for some threshold σ. For the Mahalanobis distance (Mahalanobis, 2018) measure used for the straggler mitigation experiments in S5, we calculate an average and standard deviation over B_r, i.e., $\mu_w = \mathbb{E}_{w \sim B_r}[w]$ and $\Sigma_w = \mathbb{E}_{w \sim B_r}[(w - \mu_w)^2]$, and define $\omega(w, B_r) := (w - \mu_w)^T \Sigma_w (w - \mu_w) > \sigma^2$ for some threshold σ.

C.3 OOD Sampling

Finally when want to sample a batch of size b from $W(B_a, B_r)$, forming the full set $W(B_a, B_r)$ and then sampling randomly from it has a computational cost that scales with $|B_a|$. To avoid this cost, we instead sample s experiences from B_a and keep the ones that are OOD. If this results in at least b OOD experiences, we return the samples. If not, we conclude that there aren't enough OOD samples in B_a with respect to B_r. A pseudo-code is provided in Algorithm 3.

Analytically we can model the sampling with a binomial distribution, where $p = \mathbb{E}_{z \sim B_a}[\omega(z, B_r)]$ is the fraction of samples in B_a that are OOD with respect to B_r. We will successfully get a batch of OOD samples with the probability $1 - F(b; s, p)$ where $F(\cdot; \cdot, \cdot)$ is the binomial cumulative distribution function. In all experiments in §5 we have set $s = 5b$. With $b = 200$, the success probability is 5% when $p = 0.18$ and 94% when $p = 0.22$. In other words, if at least 22% of the samples in B_a are OOD, we are highly likely to be able to collect a batch of size b of OOD samples, and unlikely if the rate is 18% or below.

Algorithm 3 OOD Sampler

Input: All samples buffer B_a, Recent samples buffer B_r, OOD function $\omega(z, B_r)$, s max sample count, b batch size
Initialize empty list $B_{out} \leftarrow \{\}$
$i \leftarrow 0$
while $|B_{out}| < b$ and $i < s$ **do**
  $z \sim B_a$ Sample from B_a
  if $\omega(z, B_r)$ Sample is OOD **then**
    $B_{out} \leftarrow B_{out} + \{z\}$ Add sample to list
  $i \leftarrow i + 1$ Increment i
if $|B_{out}| = b$ **then**
  return B_{out}
else
  return $\{\}$

Appendix D Baselines

D.1 Online EWC

EWC (Kirkpatrick et al., 2017) regularizes online learning with a Bayesian approach assuming task indices, and online EWC (Chaudhry et al., 2018; Schwarz et al., 2018) generalizes it to task boundaries. To adapt online EWC to our problem, we update the importance vector and average weights using a weighted moving average in every time step. The underlying learning approach is SAC (Haarnoja et al., 2018b).

EWC applies a regularization loss

$$\mathcal{L}_{ewc} = \alpha \sum_{k=1}^{N} ||\theta_t - \theta_k^*||^2_{F_k} \tag{14}$$

to the training loss, where N are the number of tasks, θ_k^* is the converged parameter set for task k and F_k is the diagonal of the Fisher Information Matrix (FIM) of task k on the converged model for task k. Online EWC applies an approximation of this regularization loss

$$\mathcal{L}_{ewc} = \alpha ||\theta_t - \theta_{t-1}^*||^2_{F_{t-1}^*} \tag{15}$$

using a running average F_t^* of the diagonal of the Fisher Information Matrix (FIM), and a running average of the parameters θ_t^*. The running average F_t^* is updated with a weighted average $F_t^* = (1-\beta)F_{t-1}^* + \beta F_t$, where F_t is the diagonal of the FIM respective to the recent parameters and samples.[1] Similarly, the running average θ_t^* uses the same parameter β.

Online EWC may constrain the policy output to remain constant on samples in the last $\approx \beta^{-1}$ epochs, but it has to strike a balance between how fast the importance metrics are updated with the newest FIM (larger β) and how long the policy has to remember its training (smaller β). This balance will ultimately depend on the context trace and how frequently they evolve. We tuned α and β on Pendulum-v1 for all contexts, trying $\alpha \in \{0.05, 0.1, 0.5, 1.0, 10, 100, 1000\}$ and $\beta^{-1} \in \{1M, 3M, 10M\}$ (M denotes 1 million). The returns are visualized in Figure 6 with full details in Table 2. There is no universal β that works well across all contexts and online EWC would not perform better than LCPO even if tuned to each context trace individually. We ultimately chose $\beta^{-1} = 3M$ samples to strike a balance across all contexts, but it struggled to even surpass SAC on other environments.

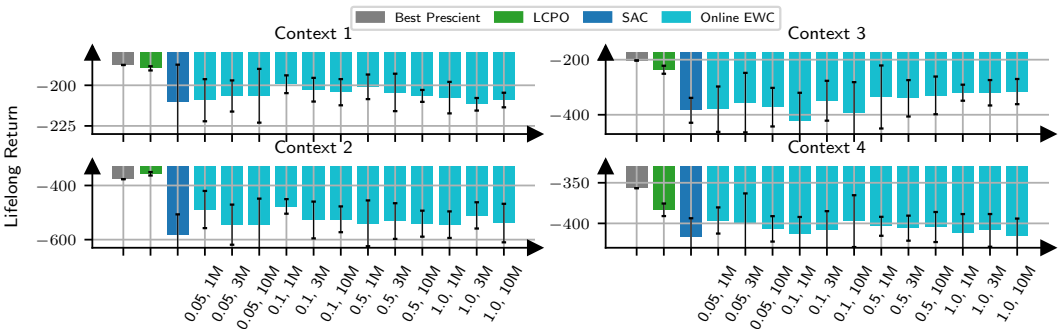


Figure 6: Pendulum-v1 lifelong returns and 95% confidence bounds of Online EWC with 12 hyperparameter trials. Hyperparameters are labeled as $\{\alpha, \beta^{-1}\}$, where α is the regularization strength and β is the averaging weight. The optimal online EWC hyperparameter is sensitive to the context, but LCPO is better even if online EWC is tuned per context.

D.2 Sliding OGD

OGD (Farajtabar et al., 2019) projects gradients for new tasks to vector spaces that are orthogonal to previous tasks' loss functions. OGD needs task labels, and Woo et al. (2022) circumvent this by making projecting.

[1] We deviate from the original notations $\{\lambda, \gamma\}$ (Rusu et al., 2016), since they could be confused with the MDP discount factor γ and GAE discount factor λ (Schulman et al., 2018).

Table 2: Average and 95th percentile confidence ranges for lifelong returns for online EWC variants in the Pendulum-v1 environment with external wind processes.

	Online EWC											
	$\alpha = 0.05$			$\alpha = 0.1$			$\alpha = 0.5$			$\alpha = 1.0$		
β^{-1}:	$1M$	$3M$	$10M$	$1M$	$3M$	$10M$	$1M$	$3M$	$10M$	$1M$	$3M$	$10M$
Context Trace 1	-209 ±13.0	-207 ±9.59	-206 ±16.6	-199 ±5.47	-203 ±7.28	-204 ±8.06	-201 ±7.57	-204 ±11.5	-207 ±3.67	-208 ±9.69	-212 ±3.87	-209 ±4.49
Context Trace 2	-489 ±68.6	-545 ±74.1	-544 ±95.3	-477 ±26.9	-527 ±67.9	-525 ±47.6	-540 ±84.6	-531 ±65.9	-541 ±48.0	-545 ±49.0	-510 ±48.5	-539 ±71.2
Context Trace 3	-380 ±82.4	-356 ±108	-372 ±70.0	-422 ±102	-349 ±72.4	-393 ±111	-335 ±114	-340 ±66.1	-330 ±68.5	-320 ±29.1	-320 ±46.1	-316 ±45.7
Context Trace 4	-396 ±16.1	-400 ±36.7	-407 ±15.6	-413 ±21.1	-408 ±23.0	-397 ±31.9	-404 ±11.7	-406 ±15.4	-404 ±18.5	-412 ±23.5	-409 ±20.1	-416 ±21.7

OGD (Farajtabar et al., 2019) circumvents CF by applying parameter updates that are orthogonal to the losses of past tasks. To do this, after a task has finished training, OGD calculates the gradient vector w.r.t. to samples for that task and stores those gradients. When the next task training begins, gradient updates are projected to orthogonal spaces w.r.t. the saved vectors from before. This procedure requires task labels and convergence. Sliding OGD (Woo et al., 2022) avoids needing task labels by using the gradient updates in the past N episodes for projection. In other words, Sliding OGD implicitly assumes that each of the past N episodes were "tasks" that have already finished training. This assumption is incorrect in our problem setup, as the context trace may change slowly. As a result, the gradient vectors of past episodes belong to the same "task" as the current episodes, and this projection fully hinders training.

D.3 Benna Fusi DQN

This approach applies a biologically plausible model for synapses on neural network weights in a deep Q network (Kaplanis et al., 2018). Conceptually, the weights are regularized with their past values in multiple different time scales. Kaplanis et al. (2018) note that while the Benna-Fusi DQN architecture was successful in simple environments, it failed with more complex and challenging ones. In our experience, this architecture did not provide any benefits compared to vanilla DDQN.

D.4 MBCD

This work handles piece-wise stationary environments by inferring change-points and task labels (Alegre et al., 2021). It trains models to predict environment state transitions, and launches new policies when the current model is inaccurate in predicting the state trajectory based on the CUSUM algorithm (PAGE, 1954). The underlying learning approach is SAC (Haarnoja et al., 2018b).

MBCD's sensitivity for detecting environment changes is a tunable hyperparameter; we tuned it by trying 6 values in a logarithmic space spanning 10^1 to 10^6 on the evaluation context traces with Pendulum-v1, and chose the best performing hyperparameter on the test set. MBCD still endlessly spawned new policies for other environments, and therefore we limited the maximum number of models to 7. Despite this, MBCD fails to perform well over the diverse set of contexts. MBCD struggles to tease out meaningful task boundaries. In some experiments, it launches anywhere between 3 to 7 policies just by changing the random seed. This observation is in line with MBCD's sensitivity in §4.1.

D.5 MBPO

MBPO (Janner et al., 2021) is a model-based approach that trains an ensemble of experts to learn the environment model, and generates samples for an SAC (Haarnoja et al., 2018b) algorithm. If the model is accurate, it can fully replay prior contexts, thereby avoiding catastrophic forgetting.

MBPO performed poorly. If the fault is the accuracy of the learned environment models, it could be improved with approaches such online meta-learning (Finn et al., 2019) or goal-

oriented model-based learning (Pong et al., 2020).. We investigated the learned models for the Pendulum-v1 experiments manually. We found these models to be very accurate, since the environment dynamics are simple.

To concretely verify that the accuracy of MBPO learned models is not the reason it underperforms, we instantiated an MBPO agent with access to the ground truth environment dynamics and context traces (but not future context traces), which we call *Ideal MBPO*. We compare the performance of Ideal MBPO vs. standard MBPO in Table 3. The performance of these two agents is widely similar. This confirms the fact that the learning algorithm itself is the problem, and not the learned models.

Table 3: Average and 95th percentile confidence ranges for lifelong returns for different algorithms and conditions in the Pendulum-v1 environment with external wind processes. An MBPO agent with access to the ground truth model performs similarly to the MBPO model. Schemes with superiority beyond 95% confidence are highlighted in bold.

	Online Learning				
	LCPO	MBPO	Ideal MBPO	A2C	**Best Prescient**
Context Trace 1	**-190 ±0.45**	-325 ±47.2	-320 ±44.2	-201 ±3.92	-187 (SAC)
Context Trace 2	**-355 ±2.57**	-843 ±24.2	-870 ±56.4	-377 ±8.05	-376 (DDQN)
Context Trace 3	**-240 ±4.41**	-603 ±213	-637 ±90.0	-307 ±32.6	-203 (SAC)
Context Trace 4	**-378 ±3.02**	-553 ±90.9	-555 ±84.1	-399 ±7.76	-357 (SAC)

The reason is the way that MBPO samples experiences for training. At every iteration, MBPO samples a batch of actual interactions from its experience buffer and generates hypothetical interactions from them. These hypothetical interactions amass in a second buffer, which is used to train an SAC agent. During the course of training, the second buffer accumulates more interactions generated from samples from the start of the experiment compared to recent samples. This is not an issue when the problem is stationary, but in non-stationary RL subject to an context process, this leads to over-emphasis of the context processes encountered earlier in the experiment. As such, MBPO fails to even surpass SAC. Prior work has observed the sensitivity of off-policy approaches to such sampling strategies (Isele & Cosgun, 2018; Hamadanian et al., 2022).

D.6 CLEAR

This approach aims to mitigate CF with off-policy learning and maintain quick adaptation with on-policy learning (Rolnick et al., 2019). They use IMPALA and V-trace (Espeholt et al., 2018) on recent batches for on-policy and stale batches for off-policy RL.

CLEAR (Rolnick et al., 2019) aims to mix the quick adaptation of on-policy RL and the CF resilience of off-policy RL, but in practice this fusion makes it very hyperparameter sensitive. The V-trace algorithm was originally intended to correct for lagging policies in a distributed RL architecture (Espeholt et al., 2018). V-trace uses clipped importance sampling to correct for the drift between the logging and training policies, which reduces variance but biases the loss. With small lags between workers, the bias is small. If V-trace is used in a scenario where the logging and training policy are very different, such as in CLEAR (Rolnick et al., 2019), the bias becomes significant enough to hinder training. CLEAR attempts to circumvent this by inducing a regularization loss on the actor and critic. Yet, this regularization will count against improving the RL policy for better returns, and will be brittle. The correct balance between policy improvement and this regularization will ultimately depend on the environment and context trace. While we tuned CLEAR extensively for the Pendulum-v1 environment, as we did for all baselines, it fails catastrophically in other environments.

D.7 PT-DQN

PT-DQN (Anand & Precup, 2023) learns two separate networks with two different goals; (1) a permanent network that is updated infrequently and slowly, and aims to learn a generalized estimate of Q-values in various tasks, and (2) a transient network that learns *and forgets* aggressively, and aims to quickly learn the optimal policy for the current task. Anand & Precup (2023) prove PT-DQN asymptotically converges to the optimum predictors in piece-wise stationary prediction tasks with tabular input spaces.

Note that PT-DQN avoids catastrophic forgetting indirectly by slowly updating a permanent Q-network. The hope is that it strikes the right balance in learning this network slowly enough such that earlier contexts are not forgotten, but updates it frequently enough such that new knowledge is not lost. This trade-off is brittle; how fast the transient Q-network should forget and relearn, and how slowly the permanent Q-network should be updated highly depends on (1) how quickly the context process changes, and (2) by how much these changes affect the transition dynamics of the MDP. As was also observed with CLEAR (§D.6), balances of this nature are brittle, due to the indirect nature of how these techniques address catastrophic forgetting.

Thus, PT-DQN is understandly hyperparameter sensitive. Indeed, to tune PT-DQN on Pendulum-v1 environment, (as done for all baselines), we carried out three rounds of grid-search on five hyperparameters, totalling 440 different combinations. Despite PT-DQN being competitive with DDQN on Pendulum-v1, the performance is unpredictable in other environments.

D.8 Off-policy RL

Off-policy RL is potentially capable of overcoming CF due to replay buffers, at the cost of unstable training. We consider DDQN (Hasselt et al., 2016) and SAC (with automatic entropy regularization, similar to LCPO) (Haarnoja et al., 2018b).

D.9 On-policy RL

On-policy RL is susceptible to CF, but more stable in online learning compared to off-policy RL algorithms, and the fast adaptation of these algorithms can also help them 'track' the optimal policy as the environment changes (Sutton et al., 2007). We compare with A2C (Mnih et al., 2016) and TRPO (single-path) (Schulman et al., 2015), with GAE (Schulman et al., 2018) applied. Note that TRPO vine is not possible in online learning, as it requires rolling back the environment world.

D.10 Prescient RL

To establish an upper-bound on the best performance that an online learner can achieve, we train *prescient policies*, as discussed in §2. We allow prescient policies to have unlimited access to the contexts and environment dynamics, i.e. they are able to replay any particular environment and context as many times as necessary. Since prescient policies can interact with multiple contexts in parallel during training, they do not suffer from CF. In contrast, all other baselines (and LCPO) are only allowed to experience contexts sequentially as they occur over time and must adapt the policy on the go. We report results for the best of four prescient policies with the following model-free algorithms: A2C (Mnih et al., 2016), TRPO (single-path) (Schulman et al., 2015), DDQN (Hasselt et al., 2016) and SAC (Haarnoja et al., 2018b).

Appendix E LCPO variants

In §4.2 we discussed our main approach to solving the constrained optimization problem below:

$$\begin{aligned} \min_{\theta} \;\; & \mathcal{L}_{PG}(\theta;B_r) \\ s.t. \;\; & D_{KL}(\theta_0,\theta;W(B_a,B_r)) \leq c_{anchor} \end{aligned} \tag{16}$$

The goal is to optimize the policy gradient loss, i.e. maximize returns on current input, while minimizing policy change on past observed inputs with the *anchoring constraint*. We outlined a solution to this problem, with a pseudo-code in Algorithm 1, that is essentially a second order constrained optimization plus a line search phase:

$$\begin{aligned} &\min_{\theta} \ (\theta-\theta_0)^T v_{PG} \\ &s.t. \ (\theta-\theta_0)^T A(\theta-\theta_0) \leq c_{anchor} \end{aligned} \tag{17}$$

where $A_{ij} = \frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j} D_{KL}(\theta_0,\theta;W(B_a,B_r))$, and $v_{PG} = \nabla_\theta \mathcal{L}_{PG}(\theta;\cdot)|_{\theta_0}$. However, there is another way to solve this problem, that we discuss as follows.

E.1 LCPO-P

An alternative way to uphold the anchoring constraint is to directly add it as term in the loss function. Let us define:

$$\begin{aligned} &\mathcal{L}_{anchor}(\theta,\theta_0;B_r,B_a) = \\ &\quad \mathbb{E}_{s,z\sim W(B_a,B_r)}[CELoss(\pi_{\theta_0}(s,z),\pi_\theta(s,z))] \end{aligned} \tag{18}$$

Where we use the Cross Entropy loss to incentivize policy anchoring. Then, we optimize the following total loss:

$$\min_{\theta} \ \mathcal{L}_{PG}(\theta;\cdot) + \kappa.\mathcal{L}_{anchor}(\theta,\theta_0;B_r,B_a) \tag{19}$$

This approach is even less compute intensive than LCPO, but is not possible in vanilla policy gradient. This is because the gradient direction from $\mathcal{L}_{anchor}$ is zero when $\theta = \theta_0$ and will not affect the optimization. Therefore, we have to repeat the gradient update several times before this term has an effect. The optimization setup in Proximal Policy Optimization (PPO) (Schulman et al., 2017) allows for several gradient steps with one batch of data, and therefore we apply the above loss in the PPO optimization step. We dub this approach LCPO-P (P stands for proximal).

Table 4: Average and 95th percentile confidence ranges for lifelong returns for LCPO and LCPO-P and conditions in the Pendulum-v1 environment with external wind processes. Schemes with superiority beyond 95% confidence are highlighted in bold.

	LCPO	LCPO-P	Best Baseline	**Best Prescient**
Context Trace 1	**-190 ±0.45**	-211 ±3.92	-201 (A2C) ±3.92	-187 (SAC)
Context Trace 2	**-355 ±2.57**	-388 ±7.33	-377 (A2C) ±8.05	-376 (DDQN)
Context Trace 3	**-240 ±4.41**	-271 ±21.9	-262 (CLEAR) ±2.2	-203 (SAC)
Context Trace 4	**-378 ±3.02**	-395 ±9.32	-399 (A2C) ±7.76	-357 (SAC)

We compare LCPO and LCPO-P in Table 4. We tuned κ on the Pendulum-v1 environment, similar to all baselines, and finalized on $\kappa = 10$. Despite this, LCPO-P fail to outperform the best baseline for each context trace even on Pendulum-v1. The stark contrast between LCPO and LCPO-P is due to the difficulty in tuning κ. LCPO's optimization constraint guarantees that the policy is anchored on past contexts, while the loss term added in LCPO-P motivates for this change to be small. Although the setup in LCPO-P is the Lagrangian dual of the setup in LCPO, they are only equivalent when κ is tuned properly per each gradient step.

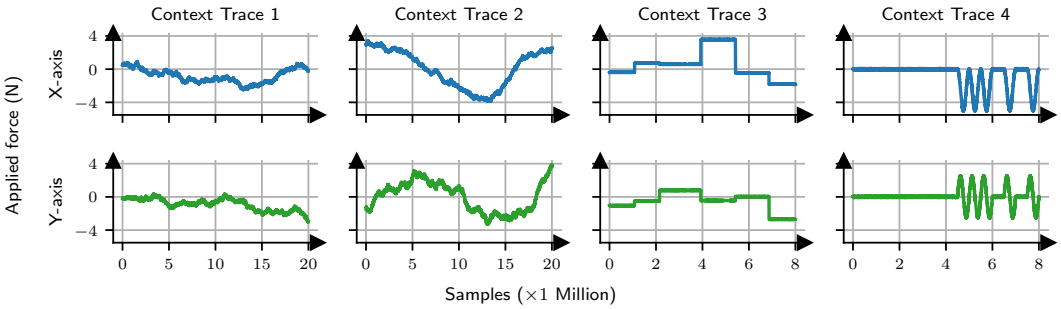


Figure 7: External wind force, per axis and context trace.

Appendix F Locomotion tasks in Gymnasium

F.1 Full results

Table 5 presents the lifelong return for all agents, environments and context traces. Table 6 presents the lifelong return for multiple LCPO agents with different OOD thresholds σ. Table 7 presents the lifelong return for multiple LCPO agents with different buffer sizes n_b.

Table 5: Average and 95th percentile confidence ranges for lifelong return for different algorithms and conditions in environments with external wind processes. Schemes with superiority beyond 95% confidence are highlighted in bold.

		Online Learning												Prescient Policies			
		LCPO	Online EWC	Sliding OGD	CLEAR	BFDQN	MBCD	MBPO	A2C	TRPO	PT-DQN	DDQN	SAC	A2C	TRPO	DDQN	SAC
Pendulum	Context Trace 1	-190 ±0.45	-204 ±2.37	-1195 ±4.38	-215 ±0.42	-346 ±15.0	-207 ±23.4	-325 ±47.2	-201 ±3.92	-222 ±4.14	-205 ±2.80	-219 ±8.88	-207 ±4.12	-208	-197	-188	**-187**
	Context Trace 2	**-355 ±2.57**	-525 ±22.6	-1232 ±3.25	-429 ±1.58	-709 ±64.9	-577 ±71.3	-843 ±24.2	-377 ±8.05	-548 ±41.3	-835 ±120	-756 ±103	-636 ±62.3	-407	-399	**-376**	-380
	Context Trace 3	**-240 ±4.41**	-345 ±29.6	-1181 ±3.78	-262 ±2.20	-766 ±75.1	-349 ±28.9	-603 ±213	-307 ±32.6	-511 ±63.4	-596 ±79.5	-612 ±76.2	-402 ±31.0	-224	-320	-228	**-203**
	Context Trace 4	**-378 ±3.02**	-412 ±6.52	-1233 ±4.41	-411 ±2.45	-693 ±53.3	-430 ±15.4	-553 ±90.9	-399 ±7.76	-648 ±31.4	-436 ±32.0	-409 ±7.59	-418 ±5.52	-383	-381	-384	**-357**
Inverse Pendulum	Context Trace 1	18.7 ±2.85	3.47 ±0.32	2.21 ±0.01	2.28 ±0.10	5.79 ±1.13	6.17 ±6.10	2.88 ±1.78	7.57 ±2.31	4.61 ±1.84	13.6 ±2.27	12.7 ±1.38	3.97 ±0.67	24.4	**25.7**	24.1	24.8
	Context Trace 2	**5.31 ±0.54**	1.55 ±0.03	1.58 ±0.00	1.60 ±0.03	1.74 ±0.09	2.10 ±1.06	1.52 ±0.01	2.36 ±0.45	1.73 ±0.28	2.24 ±0.13	1.60 ±0.06	1.59 ±0.06	7.86	**8.51**	7.18	7.75
	Context Trace 3	15.4 ±0.22	5.56 ±0.88	4.24 ±0.02	2.45 ±0.36	4.28 ±1.58	14.2 ±0.77	14.4 ±0.23	15.0 ±0.46	7.94 ±2.36	15.0 ±0.17	14.9 ±0.14	14.7 ±0.05	15.5	**16.5**	15.1	14.3
	Context Trace 4	92.4 ±0.19	5.43 ±0.88	4.64 ±0.01	2.87 ±0.17	79.6 ±2.88	46.5 ±4.85	37.6 ±0.98	**93.7 ±0.21**	75.9 ±10.6	75.5 ±3.18	92.1 ±1.18	50.5 ±1.46	102	**112**	53.7	40.1
Inverse Double Pendulum	Context Trace 1	114 ±3.54	38.8 ±2.15	38.5 ±0.10	34.2 ±0.46	48.7 ±7.99	77.3 ±13.5	68.5 ±19.3	118 ±9.47	86.3 ±13.9	76.6 ±1.91	80.0 ±0.95	75.6 ±5.67	142	**165**	88.0	95.5
	Context Trace 2	56.2 ±2.28	30.1 ±1.03	30.9 ±0.03	28.9 ±0.20	25.6 ±0.08	44.2 ±3.27	37.2 ±9.92	54.3 ±1.72	30.0 ±3.60	43.6 ±1.52	42.4 ±1.17	38.8 ±2.37	51.4	**64.0**	46.8	49.5
	Context Trace 3	65.1 ±0.98	36.1 ±1.80	36.4 ±0.06	27.1 ±0.66	39.0 ±4.49	45.8 ±2.38	42.7 ±1.55	63.4 ±2.37	49.7 ±3.93	61.9 ±0.29	61.6 ±0.26	46.8 ±0.23	63.5	**74.0**	63.6	47.4
	Context Trace 4	89.2 ±0.33	37.7 ±3.22	41.4 ±0.07	29.3 ±0.80	67.6 ±4.37	75.0 ±4.65	59.5 ±2.86	87.9 ±0.53	75.9 ±1.85	93.9 ±0.16	93.9 ±0.18	78.7 ±0.63	91.9	**96.9**	90.7	78.0
Hopper	Context Trace 1	**244 ±5.58**	34.1 ±5.05	11.4 ±0.42	3.92 ±0.54	93.5 ±3.24	153 ±33.2	130 ±12.8	203 ±7.73	164 ±7.25	86.0 ±13.9	136 ±8.83	154 ±14.1	**240**	215	155	184
	Context Trace 2	**144 ±3.44**	28.2 ±4.98	12.0 ±0.53	2.87 ±0.25	23.2 ±6.85	76.0 ±15.3	66.0 ±18.2	97.9 ±3.07	65.8 ±9.11	30.0 ±7.43	68.1 ±4.11	81.7 ±5.69	128	**131**	86.6	96.7
	Context Trace 3	**508 ±2.54**	27.0 ±10.4	19.3 ±0.60	3.62 ±0.31	137 ±12.2	240 ±81.7	264 ±49.9	484 ±13.8	443 ±17.1	52.2 ±6.63	220 ±6.15	360 ±39.2	**520**	492	327	308
	Context Trace 4	**296 ±2.59**	22.2 ±8.47	12.8 ±0.38	2.94 ±0.10	105 ±6.14	127 ±27.3	205 ±37.5	263 ±5.59	186 ±14.9	117 ±13.7	161 ±8.02	258 ±14.4	**319**	297	189	189
Reacher	Context Trace 1	-26.5 ±0.21	-16.3 ±0.60	-6622 ±124	-1780 ±958	-9.41 ±0.15	-169 ±122	-1862 ±2255	-118 ±45.9	-2346 ±471	-2492 ±1898	-7.84 ±0.03	-370 ±169	-8.00	-7.59	**-7.59**	-14.3
	Context Trace 2	-19.8 ±0.11	-16.4 ±0.37	-6733 ±74.0	-1695 ±731	-11.5 ±3.52	-339 ±493	-4308 ±3764	-102 ±34.6	-4728 ±515	-19.9 ±17.5	-7.78 ±0.07	-618 ±290	-8.33	-3866	**-7.41**	-15.3
	Context Trace 3	-27.3 ±0.89	-15.4 ±0.23	-6722 ±131	-799 ±161	-10.1 ±0.32	-596 ±857	-2058 ±4173	-133 ±61.5	-71.7 ±80.0	-5916 ±2335	-8.56 ±0.04	-258 ±140	-9.63	-8.42	**-8.41**	-15.7
	Context Trace 4	-29.4 ±0.29	-16.6 ±0.44	-6718 ±102	-1392 ±1060	-9.17 ±0.45	-121 ±161	-2354 ±2988	-117 ±56.3	-1780 ±272	-23.2 ±10.4	-9.21 ±1.22	-157 ±59.6	-9.63	-8.57	**-7.97**	-15.7

Table 6: Average and 95th percentile confidence ranges for lifelong return in LCPO with different OOD metrics and thresholds and other agents for different conditions in environments with external wind processes. L2 stands for the L2-distance OOD metric and MHD stands for the Mahalanobis distance OOD metric. Schemes with superiority beyond 95% confidence are highlighted in bold (LCPO agents are only compared against baselines).

		LCPO (L2)					**LCPO (L2)**			
		$\sigma^2=0.25$	$\sigma^2=0.5$	$\sigma^2=1$	$\sigma^2=2$	$\sigma^2=4$	$\sigma^2_{\text{MHD}}=6$	$\sigma^2_{\text{MHD}}=12$	Best Baseline	Best Prescient
Pendulum	Trace 1	-189 ±0.60	-190 ±0.76	-190 ±0.45	-195 ±0.72	-199 ±0.38	**-188 ±0.96**	-188 ±0.97	-201 (A2C) ±3.92	-187 (SAC)
	Trace 2	-349 ±2.37	-355 ±2.99	-355 ±2.57	-358 ±3.12	-361 ±3.75	-345 ±2.07	**-343 ±2.54**	-377 (A2C) ±8.05	-376 (DDQN)
	Trace 3	-241 ±5.93	-241 ±5.93	-240 ±4.41	-235 ±4.42	-232 ±5.12	-235 ±9.43	**-232 ±8.16**	-262 (CLEAR) ±2.20	-203 (SAC)
	Trace 4	-377 ±2.88	-379 ±4.59	-378 ±3.02	-381 ±2.99	-386 ±3.29	**-374 ±3.06**	-375 ±5.85	-399 (A2C) ±7.76	-357 (SAC)
Inverse Pendulum	Trace 1	**21.3 ±2.17**	18.1 ±3.76	18.7 ±2.85	5.56 ±2.01	2.79 ±0.19	3.90 ±4.76	16.4 ±6.45	13.6 (PT-DQN) ±2.27	25.7 (TRPO)
	Trace 2	4.89 ±0.66	4.27 ±0.83	5.31 ±0.54	5.62 ±0.53	2.77 ±0.83	4.47 ±1.55	**5.73 ±1.02**	2.36 (A2C) ±0.45	8.51 (TRPO)
	Trace 3	14.9 ±0.43	15.6 ±0.15	15.4 ±0.22	15.3 ±0.26	15.3 ±0.26	**16.3 ±0.06**	16.3 ±0.05	15.0 (A2C) ±0.46	16.5 (TRPO)
	Trace 4	94.6 ±0.29	92.3 ±0.22	92.4 ±0.19	92.3 ±0.22	92.3 ±0.22	**100 ±0.56**	97.8 ±0.55	93.7 (A2C) ±0.21	112 (TRPO)
Inverse Double Pendulum	Trace 1	118 ±6.30	118 ±6.30	114 ±3.54	118 ±6.30	118 ±6.30	221 ±11.0	**221 ±11.7**	118 (A2C) ±9.47	165 (TRPO)
	Trace 2	52.2 ±3.87	53.6 ±1.65	56.2 ±2.28	54.0 ±2.38	54.0 ±2.38	63.3 ±5.65	**67.3 ±9.13**	54.3 (A2C) ±1.72	64.0 (TRPO)
	Trace 3	65.4 ±1.17	64.9 ±1.14	65.1 ±0.98	64.8 ±1.14	64.8 ±1.14	60.2 ±7.87	59.0 ±7.49	63.4 (A2C) ±2.37	74.0 (TRPO)
	Trace 4	90.0 ±0.19	89.4 ±0.13	89.2 ±0.33	89.4 ±0.13	89.4 ±0.13	90.1 ±0.27	89.7 ±0.23	**93.9** (DDQN) **±0.18**	96.9 (TRPO)
Hopper	Trace 1	247 ±7.45	251 ±10.5	244 ±5.58	234 ±3.17	220 ±3.68	**253 ±22.9**	244 ±7.91	203 (A2C) ±7.73	240 (A2C)
	Trace 2	145 ±3.73	143 ±3.53	144 ±3.44	139 ±3.43	138 ±4.05	145 ±7.29	**147 ±6.58**	97.9 (A2C) ±3.07	131 (TRPO)
	Trace 3	519 ±4.76	512 ±2.91	508 ±2.54	509 ±2.91	509 ±2.91	535 ±15.2	**535 ±14.2**	484 (A2C) ±13.8	520 (A2C)
	Trace 4	**301 ±3.20**	297 ±3.38	296 ±2.59	292 ±2.39	291 ±5.21	299 ±7.71	300 ±5.25	263 (A2C) ±5.59	319 (A2C)
Reacher	Trace 1	-26.8 ±0.19	-26.5 ±0.21	-26.5 ±0.21	-24.6 ±0.37	-21.6 ±0.22	-25.2 ±0.22	-25.2 ±0.15	**-7.84** (DDQN) **±0.03**	-7.59 (DDQN)
	Trace 2	-25.5 ±0.12	-22.6 ±0.15	-19.8 ±0.11	-19.8 ±0.13	-19.8 ±0.13	-24.1 ±0.30	-23.8 ±0.48	**-7.78** (DDQN) **±0.07**	-7.41 (DDQN)
	Trace 3	-26.7 ±1.33	-26.7 ±1.33	-27.3 ±0.89	-28.9 ±0.79	-28.4 ±0.80	-26.1 ±0.99	-25.8 ±0.97	**-8.56** (DDQN) **±0.04**	-8.41 (DDQN)
	Trace 4	-30.6 ±0.32	-30.3 ±0.43	-29.4 ±0.29	-27.7 ±0.44	-26.0 ±0.40	-24.5 ±0.42	-24.1 ±0.50	**-9.17** (BFDQN) **±0.45**	-7.97 (DDQN)

F.2 Computational Load

LCPO is about 1.5× more computationally demanding than the leading baseline A2C. Table 8 depicts the total runtime per each environment interaction in the experiments in §5.1.

F.3 Action Space

Pendulum-v1 and Mujoco environments by default have continuous action spaces. We observed instability while learning policies with continuous policy classes even for the prescient policies, and were concerned about how this can affect the validity of our online experiments, which are considerably more challenging. As the action space is tangent to our problem, we discretized each dimension of the action space to 15 atoms, spaced equally from the minimum to the maximum action in each dimension. This stabilized training greatly, and is not surprising, as past work (Tang & Agrawal, 2019) supports this observation. The reward metric, continuous state space and truncation and termination conditions remain unchanged.

Table 7: Average and 95th percentile confidence ranges for lifelong return in LCPO with multiple buffer sizes and other agents for different conditions in environments with external wind processes. Schemes with superiority beyond 95% confidence are highlighted in bold (LCPO agents are only compared against baselines).

		LCPO					
		$n_b = 20M$	$n_b = 200K$	$n_b = 500$	$n_b = 25$	Best Baseline	Best Prescient
Pendulum	Trace 1	-191 ±0.65	**-190 ±0.45**	-190 ±0.62	-193 ±0.99	-201 (A2C) ±3.92	-187 (SAC)
	Trace 2	-355 ±2.02	**-355 ±2.57**	-356 ±2.01	-366 ±3.57	-377 (A2C) ±8.05	-376 (DDQN)
	Trace 3	**-240 ±6.83**	-240 ±4.41	-240 ±9.10	-259 ±10.7	-262 (CLEAR) ±2.20	-203 (SAC)
	Trace 4	-379 ±3.28	**-378 ±3.02**	-379 ±3.78	-400 ±4.62	-399 (A2C) ±7.76	-357 (SAC)
Inverse Pendulum	Trace 1	18.4 ±2.54	18.7 ±2.85	**20.8 ±1.72**	14.7 ±3.48	13.6 (PT-DQN) ±2.27	25.7 (TRPO)
	Trace 2	4.69 ±0.66	**5.31 ±0.54**	4.91 ±0.63	4.61 ±0.77	2.36 (A2C) ±0.45	8.51 (TRPO)
	Trace 3	15.4 ±0.23	15.4 ±0.22	15.4 ±0.22	15.4 ±0.23	15.0 (A2C) ±0.46	16.5 (TRPO)
	Trace 4	92.3 ±0.22	92.4 ±0.19	92.4 ±0.19	92.4 ±0.19	**93.7** (A2C) **±0.21**	112 (TRPO)
Inverse Double Pendulum	Trace 1	117 ±5.30	114 ±3.54	117 ±5.29	118 ±5.26	118 (A2C) ±9.47	165 (TRPO)
	Trace 2	54.6 ±2.19	56.2 ±2.28	54.7 ±2.19	54.9 ±2.23	54.3 (A2C) ±1.72	64.0 (TRPO)
	Trace 3	64.9 ±0.90	65.1 ±0.98	65.0 ±0.92	64.9 ±0.91	63.4 (A2C) ±2.37	74.0 (TRPO)
	Trace 4	89.4 ±0.24	89.2 ±0.33	89.4 ±0.24	89.4 ±0.24	**93.9** (DDQN) **±0.18**	96.9 (TRPO)
Hopper	Trace 1	238 ±4.45	**244 ±5.58**	241 ±5.37	233 ±3.22	203 (A2C) ±7.73	240 (A2C)
	Trace 2	141 ±4.13	**144 ±3.44**	139 ±3.76	135 ±2.54	97.9 (A2C) ±3.07	131 (TRPO)
	Trace 3	**509 ±2.48**	508 ±2.54	509 ±2.42	508 ±2.46	484 (A2C) ±13.8	520 (A2C)
	Trace 4	**296 ±2.48**	296 ±2.59	291 ±4.23	279 ±5.53	263 (A2C) ±5.59	319 (A2C)
Reacher	Trace 1	-26.6 ±0.15	-26.5 ±0.21	-26.5 ±0.19	-26.4 ±0.26	**-7.84** (DDQN) **±0.03**	-7.59 (DDQN)
	Trace 2	-19.8 ±0.12	-19.8 ±0.11	-19.8 ±0.12	-19.8 ±0.12	**-7.78** (DDQN) **±0.07**	-7.41 (DDQN)
	Trace 3	-27.8 ±0.84	-27.3 ±0.89	-27.8 ±0.97	-28.1 ±0.53	**-8.56** (DDQN) **±0.04**	-8.41 (DDQN)
	Trace 4	-29.5 ±0.31	-29.4 ±0.29	-29.4 ±0.73	-29.3 ±0.83	**-9.17** (BFDQN) **±0.45**	-7.97 (DDQN)

Table 8: Average and 95% confidence interval for training time per environment interactions, in μsec.

LCPO	A2C	TRPO	DDQN	SAC	Sliding OGD	CLEAR	BFDQN	MBPO	MBCD	Online EW	PT-DQN
0.78 ±0.03	0.52 ±0.03	0.45 ±0.03	0.45 ±0.03	0.50 ±0.02	0.65 ±0.01	0.43 ±0.01	0.31 ±0.01	2.19 ±0.04	5.62 ±0.32	3.97 ±0.02	0.42 ±0.01

We provide the achieved episodic return for all baselines in §5 in Table 9, over 10 seeds for the Pendulum-v1 envioronment, which we can compare to SB3 (Raffin et al., 2021) and RL-Zoo (Raffin, 2020) reported figures. These experiments finished in approximately 46 minutes. A2C, DDQN and SAC were trained for 8000 epochs, and TRPO was trained for 500 epochs. Evaluations are on 1000 episodes. As these results show, the agents exhibit stable training with a discretized action space.

Table 9: Average episodic returns and 95th percentile confidence ranges for different algorithms in the Pendulum-v1 environment with discretized and continuous action space.

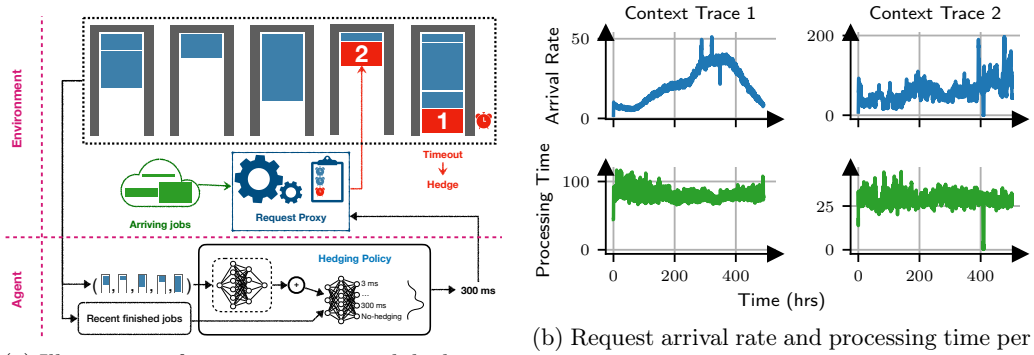
Episodic Return	A2C	TRPO	DDQN	SAC	MBPO
Discrete	-165+-5	-166+-8	-149+-2	-146+-2	-161+-3
SB3 (Raffin et al., 2021) + RL-Zoo (Raffin, 2020)	-203	-224	—	-176	—

F.4 Experiment Setup

We use Gymnasium (v0.29.1, MIT license) and Mujoco (v3.1.1, Apache-2.0 license). Our baseline and LCPO implementations use the Pytorch (Paszke et al., 2019) (v1.13.1, BSD-style license) library. Table 10 is a comprehensive list of all hyperparameters used in training and the environment.

All baselines were tuned on Pendulum-v1 via a multi-phased grid search, similar to that in §D.1. General parameters such as discount horizon were copied from the base RL algorithm each baseline is using (e.g., online EWC is using SAC, and copied SAC-specific parameters directly). Several LCPO hyperparameters were copied from TRPO, SAC and A2C (namely, entropy target, entropy learning rate, damping coefficient, rollout length, λ, γ) and the rest (c_{anchor}, c_{recent} and base entropy) were tuned with an informal search with a separate context trace (not in the evaluation set) in Pendulum-v1. The OOD threshold σ was not tuned with a search.

Appendix G Straggler mitigation



(a) Illustration of a request proxy with hedging.

(b) Request arrival rate and processing time per input.

Figure 8: Request arrival rate and processing time per input.

G.1 Full results

Figure 9 plots the tail latency across experiment time in the straggler mitigation environment for both contexts.

G.2 Experiment Setup

We use the straggler mitigation environment from prior work (Hamadanian et al., 2022), with a similar configuration except with 9 actions (timeouts of 600^{ms} and 1000^{ms} added). Similar to §F.4, our implementations of baselines and LCPO use the Pytorch (Paszke et al., 2019) (v1.13.1, BSD-style license) library. The environment code and dataset is not public and was released to us with a proprietary license. Table 11 is a comprehensive list of all hyperparameters used in training and the environment.

All baselines were tuned on a separate workload. LCPO hyperparameters were copied from the gymnasium experiments, except for base entropy which was tuned with an informal search with a separate workload (not in the evaluation set).

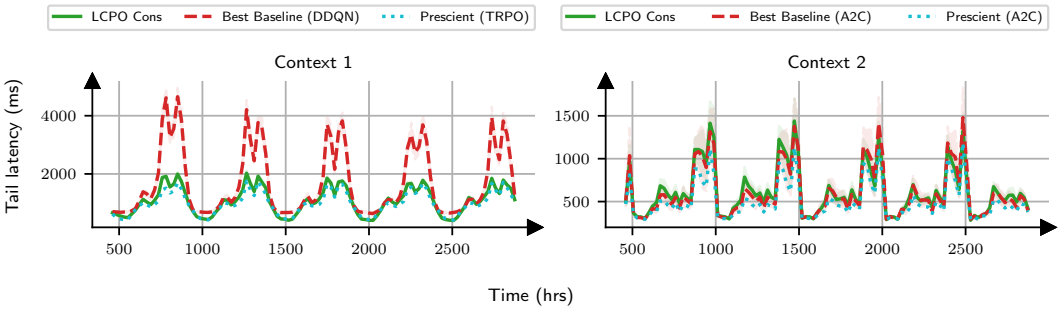


Figure 9: Tail latency with 95th percentile confidence intervals as training progresses (lower is better). We consider an initial learning period of 3.5 million samples. LCPO remains close to the prescient throughout contexts, while baselines suffer from non-stationarity.

Table 10: Training setup and hyperparameters for gymnasium environments with external wind.

Group	Hyperparameter	Value
Neural network	Hidden layers	(64, 64)
	Hidden layer activation	Relu
	Output layer activation	Actors: Softmax, Critics and DDQN: Identity mapping
	Optimizer	Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$) (Kingma & Ba, 2017)
	Learning rate	Actor: 0.0004, Critic and DDQN: 0.001
	Weight decay	10^{-4}
RL training (general)	Random seeds	25 in main experiments (§5.1), except for MBPO and MBCD 5 seeds in ablations (§5.2, §5.3, §E.1, §D.5, §D.1)
	λ (for GAE in A2C and TRPO)	0.9
	γ	0.99
A2C	Rollout per epoch	200
TRPO	Rollout per epoch	3200
	Damping coefficient	0.1
	Stepsize	0.01
DDQN	Rollout per epoch	200
	Batch Size	512
	Initial fully random period	1000 epochs
	ϵ-greedy schedule	1 to 0 in 5000 epochs
	Polyak α	0.01
	Buffer size N	All samples ($N = 20M$ or $N = 8M$)
SAC	Rollout per epoch	200
	Batch Size	512
	Initial fully random period	1000 epochs
	Base Entropy	0.1
	Entropy Target	0.1ln(15)
	Log-Entropy Learning Rate	1e-3
	Polyak α	0.01
	Buffer size N	All samples ($N = 20M$ or $N = 8M$)
LCPO	Rollout per epoch	200
	Base Entropy	0.03
	Entropy Target	0.1ln(15)
	Log-Entropy Learning Rate	1e-3
	Buffer Size n_b	1% of samples ($200K$ or $80K$)
	Damping coefficient	0.1
	c_{anchor}	0.0001
	c_{recent}	0.1
	σ	1
LCPO-P	PPO Clipping ϵ	0.2
	PPO Iterations (Max)	30
	PPO Max KL	0.01
	κ	10
MBCD	h	1000 (default was 100/300)
	max_std	3 (default was 0.5)
	N (ensemble size)	5
	NN hidden layers	(64, 64, 64)
MBPO	M (model rollouts)	512
	N (ensemble size)	5
	k (rollout length)	1
	G (gradient steps)	1

Group	Hyperparameter	Value
Online EWC	averaging weight β	0.00007 (equivalent to $\sim 3M$ samples at rollout=200)
	scaling factor α	0.1
Sliding OGD	learning rate α	1e-4
	N (window size)	1000 episodes
CLEAR	Value Clone Coefficient	1e-2
	Policy Clone Coefficient	1e-3
	Entropy Coefficient	5e-3
	V-Trace ρ	1
	V-Trace c	1
BFDQN	Depth	13
	Benna Fusi Buffer Length	2000
	g_{1_2}	1e-3
PT-DQN	Permanent Learning Rate	1e-5
	Transient Learning Rate	1e-3
	Target Update Period N	2000 steps
	Permanent Update period K	20000 steps
	Transient Forget Factor λ	0.999

Table 11: Training setup and hyperparameters for straggler mitigation experiments.

Group	Hyperparameter	Value
Neural network	Hidden layers	ϕ network: (32, 16)
		ρ network: (32, 32)
	Hidden layer activation function	Relu
	Output layer activation function	Actors: Softmax, Critics and DDQN: Identity mapping
	Optimizer	Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$) (Kingma & Ba, 2017)
	Learning rate	0.001
	Weight decay	10^{-4}
RL training (general)	Random seeds	10
	λ (for GAE in A2C and TRPO)	0.95
	γ	0.9
A2C	Rollout per epoch	4608
TRPO	Rollout per epoch	10240
	Damping coefficient	0.1
	Stepsize	0.01
DDQN	Rollout per epoch	128
	Batch Size	512
	Initial fully random period	1000 epochs
	ϵ-greedy schedule	1 to 0 in 5000 epochs
	Polyak α	0.01
	Buffer size N	All samples ($N = 21M$)
SAC	Rollout per epoch	128
	Batch Size	512
	Initial fully random period	1000 epochs
	Base Entropy	0.01
	Entropy Target	0.1ln(9)
	Log-Entropy Learning Rate	1e-3
	Polyak α	0.005
	Buffer size N	All samples ($N = 21M$)
LCPO	Rollout per epoch	128
	Base Entropy	0.01
	Entropy Target	0.1ln(9)
	Log-Entropy Learning Rate	1e-3
	Buffer Size n_b	$210K$
	Damping coefficient	0.1
	c_{anchor}	0.0001
	c_{recent}	0.1
MBCD	h	300000 (default was 100/300)
	max_std	3 (default was 0.5)
	N (ensemble size)	5
	NN hidden layers	(64, 64, 64)
MBPO	M (model rollouts)	512
	N (ensemble size)	5
	k (rollout length)	1
	G (gradient steps)	1
Online EWC	averaging weight β	0.00007 (equivalent to $\sim 2M$ samples at rollout=128)
	scaling factor α	0.1