Statistical Guarantees for Lifelong Reinforcement Learning using PAC-Bayesian Theory

${f Zhi}$ ${f Zhang}^1$	$\mathbf{Chris} \ \mathbf{Chow}^2$	
Haochen $Zhang^1$	${f Eric}$ Hanchen Jiang ¹	
	$\mathbf{Yuchen}\ \mathbf{Cui}^1$	

 $\begin{array}{ccc} {\bf Yasi \ Zhang^1} & {\bf Yanchao \ Sun^3} \\ {\bf Han \ Liu^4} & {\bf Furong \ Huang^5} \\ {\bf Oscar \ Hernan \ Madrid \ Padilla^1} \end{array}$

¹University of California, Los Angeles ²Niantic Labs ³Apple Inc. ⁴Northwestern University ⁵University of Maryland, College Park

Abstract

Lifelong reinforcement learning (RL) has been developed as a paradigm for extending single-task RL to more realistic, dynamic settings. In lifelong RL, the "life" of an RL agent is modeled as a stream of tasks drawn from a task distribution. We propose EPIC (Empirical PAC-Bayes that Improves Continuously), a novel algorithm designed for lifelong RL using PAC-Bayes theory. EPIC learns a shared policy distribution, referred to as the world policy, which enables rapid adaptation to new tasks while retaining valuable knowledge from previous experiences. Our theoretical analysis establishes a relationship between the algorithm's generalization performance and the number of prior tasks preserved in memory. We also derive the sample complexity of EPIC in terms of RL regret. Extensive experiments on a variety of environments demonstrate that EPIC significantly outperforms existing methods in lifelong RL, offering both theoretical guarantees and practical efficacy through the use of the world policy.

1 Introduction

Deep reinforcement learning (RL) has excelled in challenging tasks including abstract strategy games (Silver et al., 2017, 2016), visual navigation (Zhu et al., 2017), and control (Mnih et al., 2015; Lillicrap et al., 2015). However, RL is a data intensive learning paradigm, therefore training a policy for every task from scratch is computationally expensive and time-consuming. In reality, many tasks an agent encounters are not entirely novel but instead belong to a broader task distribution, meaning they share commonalities that can be leveraged. This insight highlights the inefficiency of retraining for every individual task. Lifelong RL, also known as continual RL, emerges as a promising framework where an agent interacts with a sequence of tasks, continuously adapting and improving its policy by leveraging knowledge from past task instances (Khetarpal et al., 2022).

In lifelong RL, an agent's objectives are primarily to achieve **fast adaptation** with limited samples and effective **knowledge retention** (Abel et al., 2024). In other words, lifelong RL agents experience a stabilityplasticity dilemma, where the agent must balance retaining useful long-term knowledge with the ability to rapidly adapt to new situations.

Recent methods addressing knowledge retention and transfer in lifelong RL include Q-value function transfer (Lecarpentier et al., 2021), optimizing Q-value function initialization (Abel et al., 2018), decomposing the value function into permanent and transient components (Anand and Precup, 2023), reusing knowledge by sampling from past experiences (Kessler et al., 2023), detecting change points in rewards and environment dynamics(Steinparz et al., 2022), and using a Bayesian approach to learn a common task distribution for better data efficiency and transfer (Fu et al., 2022).

In lifelong RL, domain shifts induce non-stationarity, which occurs not only due to changing transition dynamics and reward functions, but also through variations in available actions or decisions over time(Boutilier et al., 2018; Chandak et al., 2020). Such scenarios are common in real-world applications. For example, in robotics, additional control components are integrated throughout the robot's lifetime, and in medical decision support systems, new treatments and medications must be incorporated.

Furthermore, tasks encountered over an agent's lifetime can be highly diverse, yet certain high-level strategies that can be shared across tasks. Not only should the world model (Ha and Schmidhuber, 2018; Fu et al., 2022; Anand and Precup, 2023), which captures the general knowledge distribution of tasks, be continuously refined, but it is also crucial for an agent to develop a *world policy*. The key consideration of this *world policy* is to adapt the parameters of the policy so they are captured by a global distribution, which represents the uncertainty over policy parameters. This allows for better generalization and adaptability across tasks.

Motivated by the above need, we raise two questions:

- 1. Can we extract the common structure present in policies from previously encountered tasks, allowing the agent to quickly learn the policy specific to new task to enable fast adaptation *with* theoretical guarantees?
- 2. How many samples are required to achieve a given level of performance?

To answer these two questions, we develop a unified framework based on a Bayesian method that learns a rapidly adaptable policy distribution from past tasks, retaining valuable information while remaining capable of quickly adapting to unseen situations.

We also provide a theoretical analysis of the algorithm's generalization performance relative to the number of effective tasks and retained knowledge in the finite Markov Decision Process (MDP) setting, along with its sample complexity to demonstrate efficiency.

When addressing the first question, we must also account for both catastrophic forgetting and generalizability - the aforementioned stability-plasticity dilemma. Agents that can quickly solve traditional RL problems risk abruptly losing performance on tasks they have seen before due to their flexibility or plasticity. On the other hand, agents that do not forget any of their past experience may give up a measure of their plasticity. These issues are central in lifelong RL, and can be approached from a Bayesian perspective (Khetarpal et al., 2022). Bayesian methods have been applied to meta learning (Amit and Meir, 2018), lifelong learning for bandits (Flynn et al., 2022), and learning controls for robots in multiple environments (Majumdar et al., 2021), aiming to learn a fast adapted policy. Instead of learning a specific policy, we leverage the PAC-Bayes theory to learn a distribution of policy hypotheses shared across multiple tasks. Further details about PAC-Bayes theory can be found in Section 2.3 and the Related Works section in Appendix §B. When a new task arises, we can initialize a policy hypothesis by sampling from this learned distribution. A well-constructed distribution of hypotheses promotes effective long-term memory, mitigating catastrophic forgetting. Unlike

prior methods, we sample a random policy function according to this distribution. This function sampling approach is seen in modern popular deep learning methods, for example, in-context learning(Garg et al., 2022).

For the second question, an agent has to keep learning as well as forgetting. Too much experienced knowledge kept in memory may decrease the learning efficiency, while too little may be insufficient to learn an effective policy distribution. To address the second question, we derive a relationship between the performance of our algorithm and the number of experienced tasks (denoted as N in a later section) that need to be retained in the agent's memory, based on PAC-Bayes theory. We use the negative expected long term rewards, where the expectation is taken with respect to tasks and policies (also known as the generalization error), as a measure of the algorithm's performance from a statistical perspective.

From our theoretical result, where we provide an expression of this relationship, we discovered a trade-off between this value and the algorithm's performance, which aligns with natural intuition, a double sided effect. In practice, our expression allows us to optimize the performance of our algorithm by optimizing N, although we recommend using hyperparameter tuning. Furthermore, to demonstrate the efficiency of our algorithm, we derive its sample complexity from a RL regret perspective, showing that our algorithm learns an optimal policy as more tasks encountered.

Our Contributions. In this work, we introduce a novel PAC-Bayes framework tailored to lifelong RL, addressing critical challenges including changing decisions, catastrophic forgetting and efficient knowledge retention. Our contributions are summarized as follows:

- We propose EPIC (Empirical PAC-Bayes that Improves Continuously), a lifelong RL algorithm that leverages PAC-Bayes theory to learn a shared policy distribution, referred to as the *world policy*. This world policy enables the agent to quickly adapt to new tasks while retaining useful knowledge from past experiences, providing theoretical guarantees of generalization across tasks.
- We derive a novel PAC-Bayes bound for lifelong RL and provide a theoretical analysis that links long-term rewards to the number of retained past tasks, ensuring a balance between memory usage and performance across diverse tasks. We provide a sample complexity of our approach in terms of RL regret.
- We evaluate EPIC through extensive numerical experiments with common lifelong RL benchmarks, as well as additional environments we created. Our results show EPIC outperforms prior methods.

These results underscore EPIC's effectiveness in lifelong learning scenarios, offering a robust and theoretically grounded solution for continual adaptation in RL.

2 Preliminaries

2.1 Reinforcement Learning

In RL, an agent interacts with the environment by taking actions, observing states and receiving rewards. The environment is modeled by a Markov Decision Process (MDP), which is denoted by a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T, R, \gamma, \nu \rangle$, where \mathcal{S} is the state space, \mathcal{A} is the action space, T is the transition kernel, R is the reward function, $\gamma \in (0, 1)$ is the discount factor, and ν is the initial state distribution.

A trajectory $\tau \sim \pi$ generated by policy π is a sequence $s_1, a_1, r_1, s_2, a_2, \cdots$, where $s_1 \sim \nu, a_t \sim \pi(a|s_t)$, $s_{t+1} \sim T(s|s_t, a_t)$ and $r_t = R(s_t, a_t)$. The goal of an RL agent is to find an optimal policy π^* that maximizes the expected total rewards $J(\pi) = \mathbb{E}_{\tau \sim \pi}[r(\tau)] = \mathbb{E}_{s_1,a_1,\cdots,\nu,\pi,T,R}[\sum_{t=1}^{\infty} \gamma^{t-1}r_t].$

2.2 Lifelong Reinforcement Learning

In lifelong RL, the agent interacts with a (potentially infinite) sequence of tasks, which come from an underlying task distribution(Khetarpal et al., 2022), denoted as \mathcal{D}_i , $i = 1, ..., \infty$. Suppose that tasks share the same γ , but may have different S, \mathcal{A} , transition probabilities T and rewards R. The learning process is:

- 1. Initialize a policy π_0 ;
- 2. Sample a task (MDP) $\mathcal{M}_i \sim \mathcal{D}_i$;
- 3. Starting from π_0 , learn a policy π_i for task \mathcal{M}_i to maximize rewards.

An effective lifelong RL agent should quickly adapt to new tasks that it encounters throughout its life.

2.3 PAC-Bayes Theory

PAC-Bayes analysis applies to learning algorithms that output a distribution over hypotheses $h \in \mathcal{H}$. This refers to h is sampled independently from a distribution over functions in a function class \mathcal{H} . For example, for a linear predictor of d dimension, $h(x) = \langle w, x \rangle$, we let $w \sim \mathsf{N}(0, I_d)$. Generally, such algorithms will be given a prior distribution $\underline{P} \in \mathcal{P}$ at the beginning and learn a posterior distribution $P \in \mathcal{P}$ after observing training data samples $\{z_i\}_{i=1}^N$. We define the expected loss (generalization error) $l_{\mathcal{D}}(P) = \mathbb{E}_{h\sim P} [\mathbb{E}_{z\sim\mathcal{D}} [h(z)]]$, and the empirical loss (training error) $l_{\mathcal{S}}(P) = \mathbb{E}_{h\sim P} \left[\frac{1}{N} \sum_{i=1}^{N} h(z_i)\right]$, which are under the expectation of hypothesis $h \sim P$.

The main application of PAC-Bayes analysis in machine learning is to produce high-confidence bounds for the true or generalization error in terms of the training error plus $\mathscr{R}(\mathbb{D}_{KL}(P||\underline{P}))$, which is a function of the KL divergence between the prior and posterior distributions, as shown below (McAllester, 1999),

$$l_{\mathcal{D}}(P) \le U(P) \coloneqq l_S(P) + \mathscr{R}(\mathbb{D}_{KL}(P \| \underline{P})), \qquad (1)$$

with

$$\mathscr{R}(\mathbb{D}_{KL}(P\|\underline{P})) \\ \coloneqq \sqrt{\frac{1}{2N} \left[\mathbb{D}_{KL}\left(P\|\underline{P}\right) + \log\left(2N^{1/2}/\delta\right) \right]},$$
(2)

where U(P) in right-hand side of Equation (1) is called the generalization error bound that depends on P, and minimization of this bound leads to generalization error guarantees.

3 Methods

We propose a PAC-Bayes lifelong RL algorithm, EPIC (Algorithm 1), to minimize the novel bound in (3). The algorithm utilizes a Bayesian posterior to distill the common policy distribution learned from previous tasks, which is then used to sample the policy and serves as a prior for new tasks. We provide a generalization guarantee for EPIC in Theorem 3.3. Furthermore, we employ the Gaussian family for the posterior and prior in EPICG (Algorithm 2). The sample complexity of EPICG is given in Theorem 3.4.

3.1 PAC-Bayes Framework for Lifelong RL

We learn a general policy distribution P for lifelong RL by leveraging the core concept of the PAC-Bayes Method. We explicitly formulate U(P) for the lifelong RL setting and employ it to propose an algorithm that learn the P by minimizing U(P) to accomplish the lifelong learning objective.

Define \mathcal{P} as the whole policy space for P. Rather than considering a general distribution P for hypotheses where Π can be infinite, we let \mathcal{P} be parameterized by $\theta \in \mathbb{R}^d$ such that $\theta \sim P$. Note θ could be a neural network.

Naturally, the distribution P is the posterior distribution of policy θ in the PAC-Bayes framework. Then let P be the prior distribution of the parameter. In the lifelong setting, as the tasks stream in, assume the agent has encountered K tasks so far, then the PAC-Bayes lifelong RL problem is formulated as follows:

$$\min_{P} U(P)$$

$$\coloneqq \frac{1}{K} \sum_{i=1}^{K} \left\{ \underset{\theta \sim P}{\mathbb{E}} \left[-J_{\mathcal{M}_{i}}(\pi_{\theta}) \right] \right\} + \mathscr{R}(\mathbb{D}_{KL}(P \| \underline{P})), \qquad (3)$$

where $\mathscr{R}(\mathbb{D}_{KL}(P||\underline{P}))$ is derived later in our theory,

where $J_{\mathcal{M}}(\pi_{\theta})$ is the total expected reward of policy π in MDP \mathcal{M} , taking the expectation with respect to the posterior distribution P for the parameter θ . The negative sign can be interpreted as the loss on a specific task \mathcal{M} ,

To be concrete, in the finite MDP setting with length H, for policy π_{θ} with $\theta \sim P(\theta)$, the total expected reward with task \mathcal{M} is the value function,

$$J_{\mathcal{M}}(\pi_{\theta}) = \mathbb{E}\left[\sum_{h=1}^{H-1} \gamma^{h-1} r_h | \pi_{\theta}, s_1, \mathcal{M}\right],$$

from a length of H consecutive sample transitions, $s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_H \sim \pi_{\theta} \times \mathcal{M}.$

3.2 An Algorithm based on PAC-Bayes Lifelong Framework

We now develop an algorithm to exploit the PAC-Bayes framework to efficiently perform lifelong RL.

Consider a time where we have seen K tasks so far, and denote them $\{\mathcal{M}_i\}_{i=1}^K$. They are drawn from the lifelong task distribution $\{\mathcal{D}_i\}_{i=1}^K$. It's worth noting that these tasks can exhibit distinct distributions and interdependencies. Each distribution \mathcal{D}_i should possess non-zero support and boundedness both from above and below. Critically, once the agent interacts with a task, revisiting previously encountered MDPs is not guaranteed.

Our objective is to learn a shared lifelong learning model - the distribution of θ using the K tasks the algorithm has encountered so far.

To achieve this, we propose the following lifelong RL learning algorithm based on the learning objective in Equation (3), and provide its theoretical justification. The main idea is to learn a policy distribution P as a policy initializer using the objective in Equation (3), referred to as the *default policy* This approach allows the default policy to capture common knowledge among tasks, addressing the challenge of task divergence.

In the lifelong setting, the agent receives a new task, stores it, learns from it, then forgets. We allow the agent to keep a number of N tasks in memory. We update the default policy every N tasks and estimate the training cost based on the most recent N tasks. At the K-th task, the agent has performed $\lfloor \frac{K}{N} \rfloor$ updates to the default policy so far. At each time step $l = 1, \dots, \lfloor \frac{K}{N} \rfloor$, the agent has θ_{l-1} as its policy parameters from P_{l-1} . It encounters the *i*th task $\mathcal{M}_{l,i}$'s MDP, and receives $J_{\mathcal{M}_{l,i}}(\pi_{\theta_{l-1}})$ as the total discounted expected reward. The collects trajectory data of H steps for task $\mathcal{M}_{l,i}$, using $\pi_{\theta_{l-1}}$, resulting in a dataset $\tau_l = (\tau_{l,1}, \dots, \tau_{l,N})$ with a size of $|\tau_l| = HN$.

Algorithm 1 Empirical PAC-Bayes that Improves Continuously (EPIC)

- 1: **Input:** Update frequency N; the number of steps allowed in each task H; prior evolving speed λ
- 2: Initialize prior policy distribution \underline{P}_0
- 3: Initialize default policy distribution $P_0 \leftarrow P_0$;
- 4: for $i = 1, 2, 3, \dots, K, \dots \infty$ do
- 5: Receive a new task $\mathcal{M}_i \sim \mathcal{D}_i$ and store it into Memory buffer.
- 6: **if** $i \mod N = 0$ **then**
- 7: Let l = i/N.
- 8: Sample $\theta_{l-1} \sim P_{l-1}$
- 9: Roll out trajectories $\tau_{l,k}$ using $\pi_{\theta_{l-1}}$ and $\{\mathcal{M}_k\}_{k=i-N+1}^i$ and store τ_l into Memory.
- 10: # Update default policy P_l by using τ_l
- 11: $\underline{P}_{l-1} \leftarrow (1-\lambda)\underline{P}_{l-1} + \lambda P_{l-1}$
- 12: $P_l \leftarrow \arg\min_P U(P)$
- 13: Empty Memory by clearing dataset τ_l
- 14: **end if**
- 15: end for

The agent uses τ_l to update the default policy P_l by minimizing the generalization error bound in (3), evaluated at the current time's posterior P_{l-1} and prior P_{l-1} . Before learning starts, the agent initializes a prior policy distribution P_0 and the same posterior policy distribution P_0 randomly or based on domain knowledge (Lines 2-3 of Algorithm 1). Choosing a good prior policy distribution P_0 is challenging as it affects the tightness of the bound.

We adopt a Bayesian sequential experiment design (Chaloner and Verdinelli, 1995) and use an evolving prior instead of a fixed one. We gradually move the prior towards the default policy by $P_l = (1 - \lambda)P_l + \lambda \times P_l$ (Line 11), where $\lambda \in (0, 1)$ controls the moving speed, and λ decays by $\lambda = \lambda \times \alpha$ ($\alpha < 1$) over the tasks. This allows us to find a good prior during learning and leverage it to improve the default policy.

As the agent encounters an increasing number of tasks, each task remains distinct. However, with more exposure to tasks, the agent gradually improves its understanding of the distribution P for π_{θ} . When a new task emerges, the agent can sample $\theta_l \sim P_{l-1}$, employing θ_l to generate a trajectory for subsequent updates. This allows the agent to learn faster, obtaining higher rewards in a shorter time frame. Next, we derive our main PAC-Bayes theorem for Algorithm 1.

Our learning process involves a loop of times to evolve the policy distribution. So we index the policy distribution at each time by a subscript. First, denote $\theta := \{\theta_l\}_{l=0}^{\lfloor \frac{K}{N} \rfloor - 1}$ and let $P := P(\{\theta_l\}_{l=0}^{\lfloor \frac{K}{N} \rfloor - 1})$ denote the joint posterior distribution of $\theta_0, \ldots, \theta_{\lfloor \frac{K}{N} \rfloor - 1}$ across all times. And naturally, let $P_l := P(\theta_l | \theta_{l-1})$ be the conditional probability of policy for time l given the policy from time l-1, and specially, let $P_0 := P(\theta_0)$.

Assumption 3.1. 1) Given the previous policy θ_{l-1} , the current time policy is independent of the historical sequence:

 $\theta_l \perp \theta_{l-2}, \ldots, \theta_0 | \theta_{l-1}.$

Intuitively, this conditional independence assumption for policy implies that the current policy θ_l is conditionally independent of the past given θ_{l-1} .

2) Define the support lower bound of the policy distribution as:

$$s_{\min} = \inf \big\{ s : s \ge \min_{A: P_l(A) > 0} P_l(A) \text{ for all } P_l \in \Pi \big\}.$$

Intuitively, this is the smallest non-zero probability across all policies θ_l and measures $P_l \in \Pi$.

3) Define the norm difference of two distributions as

$$\begin{aligned} \|P_l - P_{l-1}\|_{\infty} \\ \coloneqq \sup \left\{ |P_l(A) - P_{l-1}(A)| : A \in \mathscr{A}, P_l, P_{l-1} \in \Pi \right\}. \end{aligned}$$

Then, we can define the **radius of variation** between consecutive policy distributions as:

$$r = \inf \{ c : \|P_l - P_{l-1}\|_{\infty} \le c \text{ for all } l \},\$$

where \mathscr{A} is the set of measurable events. This radius r bounds the difference between consecutive policy measures P_l and P_{l-1} .

To simplify the analysis and improve readability, we denote $T = \lfloor \frac{K}{N} \rfloor$ and assume $K \mod N = 0$ without loss of generality. Based on Assumption 3.1, we arrive at the following relationship:

$$P(\{\theta_l\}_{l=0}^{T-1}) = P_{T-1} \times \dots \times P_l \times \dots \times P_0.$$
 (4)

We also derive a corollary on the decomposition of the training error, which facilitates the subsequent proof. The proof is deferred to Appendix C.1.

Corollary 3.2 (Decomposition of Training Error). Suppose Assumption 3.1 holds. Then we have:

$$\frac{1}{K} \sum_{i=1}^{K} \mathbb{E}_{\{\theta_l\}_{l=0}^{T-1} \sim P} [-J_{\mathcal{M}_i}(\pi_{\theta})] \\ = \sum_{l=1}^{T} \sum_{i=1}^{N} \frac{1}{TN} \mathbb{E}_{\theta_{l-1} \sim P_{l-1}} \left[-J_{\mathcal{M}_{l,i}}(\pi_{\theta_{l-1}})\right].$$

Theorem 3.3 (PAC-Bayes Bound for EPIC). Under the settings of Algorithm 1 and Assumption 3.1, further assume *i*-th finite horizon MDP of task *i* has a reward that belongs to [0, 1]. When running Algorithm 1, we update the default policy distribution P_l for every *N*-th task by using pairs of $\{(P_l, P_l)\}_{l=0}^{T-1}$. Let $\mathscr{T} := \prod_{l=1}^{T} (\theta_{l-1} \times \mathcal{M}_l)$, and let the expected loss over joint policy and joint trajectory space be:

$$\frac{1}{K}\sum_{i=1}^{K} \mathbb{E}_{\{\theta_l\}_{l=0}^{T-1} \sim P}[\mathbb{E}_{\{\tau_l\}_{l=1}^{T} \sim \mathscr{T}}[-J_{\mathcal{M}_i}(\pi_{\theta})]].$$

and let the training error be:

$$\frac{1}{K} \sum_{l=1}^{T} \sum_{i=1}^{N} \mathbb{E}_{\theta_{l-1} \sim P_{l-1}} \left[-J_{\mathcal{M}_{l,i}}(\pi_{\theta_{l-1}}) \right].$$

Then with probability at least $1 - 2 \exp(-K^{\gamma})$, for any $0 < \gamma < 1$, we have

expected loss \leq training error $+ \mathscr{R}(\mathbb{D}_{KL}(P||\underline{P})),$ with

$$\mathscr{R}(\mathbb{D}_{KL}(P\|\underline{P})) = \frac{2N^{1/2}H\frac{\lambda r}{1-\alpha}\sqrt{\frac{1-\alpha^{2(K/N-1)}}{s_{\min}(1-\alpha^{2})}}}{K^{1/2}} + \frac{2N^{1/2}H}{K^{(1-\gamma)/2}}.$$
(5)

The proof is deferred to Appendix §C.

Remarks. (1) The tightness of the bound depends on the number of lifelong tasks encountered so far K, the number of tasks memorized N, the trajectory length H, and the KL divergence between P and \underline{P} . By letting $\gamma = 1/4$, the difference of training and generalization error is in the order $\mathcal{O}(K^{-3/8})$. (2) The N appears on the right hand side can be understood as the memory size kept in the agent before it refreshes. The larger N will reduce K/N thus could potentially decrease the first term by making $1 - \alpha^{2K/N-2}$ smaller. However, it also increase the value $N^{1/2}$. One could obtain a reasonable choice of N by minimizing the U(P) = Training error $+ \mathscr{R}(\mathbb{D}_{KL}(P||P))$ with respect to N, but the training error is unknown.

Practically, a strategy to adaptively adjust N by minimizing the U(P) using a neural network can work.

Overall, this theory enables our learner to optimize the right-hand side of Equation (5) and learn the lifelong policy distribution with a guaranteed minimal true cost.

There are several unsolved questions before we propose a practical lifelong RL algorithm. Theorem 3.3 holds for policy distribution P and prior distribution \underline{P} parameterized by θ . However, in practice, determining suitable distributions for \underline{P} and P becomes a crucial challenge. Additionally, computing the posterior distributions $\{P_l\}_{l=0}^{T-1}$ is non-trivial. Moreover, we need to identify the appropriate optimization method to learn parameters for P. To address these questions, the next two sections will provide solutions and propose a practical lifelong RL algorithm based on the proposed Algorithm 1.

3.3 Posterior Distribution and Prior Distribution

In Equation (5), P_l represents the posterior distribution of θ_l . To optimize the posterior P_l , we need to choose appropriate hyperparameters for its distribution. For instance, in a Gaussian distribution, we optimize its mean μ and variance σ^2 .

Let $\tau_l \sim \theta_{l-1} \times \mathcal{M}_1 \cdots \times \ldots \mathcal{M}_N$ be the data induced from previous θ_{l-1} , and define the likelihood function $p(g(\tau_l)|\theta_{l-1})$. Suppose the prior distribution is a probability density function $\underline{p}(\theta_{l-1}; q)$, parametrized by q, such as a Gaussian prior $\underline{P}_l = \mathcal{N}(\underline{\mu}_l, \underline{\sigma}_l)$, where $q \coloneqq (\underline{\mu}_l, \underline{\sigma}_l)$.

Based on Bayes' Rule, the posterior distribution is uniquely given by $p(\theta_{l-1}|g(\tau_l);q) = \frac{p(g(\tau_l)|\theta_{l-1})p(\theta_{l-1};q)}{c(q)}$, where c(q) is the normalization constant depends on q. We can optimize the hyperparameter q using the following equation:

$$\min_{q} U_{l}(P;q) \coloneqq \sum_{i=1}^{N} \mathbb{E}_{\theta_{l-1} \sim p(\theta_{l-1}|g(\tau_{l});q)} \left[-J_{\mathcal{M}_{l,i}}(\pi_{\theta_{l-1}}) \right] \\
+ \mathscr{R}(\mathbb{D}_{KL}(P \| \underline{P};q)),$$
(6)

where $\mathscr{R}(\mathbb{D}_{KL}(P||\underline{P};q))$ is defined in (5). In Equation (6), the posterior distribution is unknown, and obtaining an explicit expression requires knowing the data likelihood. In the RL regime, data samples consist of states, actions, and value functions, and one approach is to use the exponential of the negative squared temporal difference (TD) error as an unnormalized likelihood, as suggested in Dann et al. (2021), which is left for future research.

In PAC-Bayes, the prior and posterior distributions can belong to different families. However, it is often practical to consider them belonging to a common distribution family, as it simplifies the computation of KL divergence. Hence, we assume the default and prior policy distributions for θ to be *d*-dimensional Gaussians with unknown parameters $\theta \sim \mathcal{N}(\mu, \sigma^2)$. These parameters are updated by minimizing the upper bound.

Based on equation (6), we solve the following problem

where $\phi(\theta; \mu, \sigma)$ is the Multivariate Gaussian PDF:

$$\min_{\mu,\sigma} U(P;\mu,\sigma) \coloneqq \sum_{i=1}^{N} \int -J_{\mathcal{M}_{l,i}}(\pi_{\theta_{l-1}})\phi(\theta;\mu,\sigma)d\theta + \mathscr{R}\left(\mathbb{D}_{KL}\left(\mathcal{N}(\mu,\sigma^{2}) \| \mathcal{N}(\underline{\mu};\sigma_{-}^{2})\right)\right).$$
(7)

Evaluating the integral in equation (7) analytically is intractable in practice. Therefore, we resort to Monte Carlo Methods, where we sample $\theta_{l-1,j_{j}\in[M]}$ to approximate the gradient descent updates by:

$$\nabla_{\mu,\sigma} \hat{U}(P, \{\mathcal{M}_i\}_{i \in [N]}, \{\theta_{l-1,j}\}_{j \in [M]}; \mu, \sigma)$$

$$\coloneqq \frac{1}{M} \sum_{j=1}^{M} \sum_{i=1}^{N} -\nabla_{\mu,\sigma} \{J_{\mathcal{M}_{l,i}}(\pi_{\theta_{l,j}}) + \mathscr{R}\left(\mathbb{D}_{KL}\left(\mathcal{N}(\mu, \sigma^2) \| \mathcal{N}(\underline{\mu}; \sigma_{-}^2)\right)\right)\},$$
(8)

where $\theta_{l-1,j}$ is a sample drawn from P_{l-1} to perform gradient descent during optimization in each iteration.

Moreover, to ensure the parameters μ and σ can be updated, we use indirect sampling by first sampling a multivariate standard normal distribution ϵ_j . The randomness of the parameter θ_j is then defined as:

$$\theta_j = \mu + \sigma \odot \epsilon_j, \quad \epsilon_j \sim \mathcal{N}(0, I_d).$$
 (9)

According to Equation (9), the parameter θ_j is multivariate normal distributed with $\theta_j \sim \mathcal{N}(\mu, \sigma^2)$.

3.4 A Practical EPIC Algorithm

We propose a practical EPIC algorithm, called EPICG, as presented in Algorithm 2. In this algorithm, a policy is defined as a Gibbs distribution in a linear combination of features: $\pi_{\theta}(s, a) = \frac{\exp(\theta^{\top}\psi_{s,a})}{\sum_{b}\exp(\theta^{\top}\psi_{s,b})}$. Here, θ can be replaced by a neural network.

For the parameterization of θ using a neural network, we provide the details in Appendix §E.1. In each iteration, the agent samples a set of policies $\theta_{jj\in[M]} \sim P$ from the "posterior" policy distribution for every Ntasks (Lines 8-9). It then rolls out a set of trajectories $\tau = \tau_{i \times j \in [N] \times [M]}$ for each task and estimates the cost (Lines 10-11). More specifically, an action a is sampled as $a \sim \pi_{\theta_{l,j}}(s, a)$, and a state s is sampled using a transition kernel determined by task \mathcal{M}_i .

The gradient is taken with respect to the objective function \hat{U} (Equation (7)) which with respect to the cost function and with respect to the KL divergence function expressed in Equation (5).

EPICG uses gradient descent in the space of P to find the policy that minimizes the expected loss, i.e., $P^* \in$ $\arg \inf_{P \in \Pi} \frac{1}{K} \sum_{i=1}^{K} \mathbb{E}_{\{\theta_l\}_{l=0}^{T-1} \sim P} [\mathbb{E}_{\{\tau_l\}_{l=1}^{T} \sim \mathscr{T}} [-J_{\mathcal{M}_i}(\pi_{\theta})]].$



(a) HalfC.-gravity (b) HalfC.-bodyp. (c) Hopper-gravity (d) Hop.-bodyp. (e) Walker-gravity (f) Walker-bodyp.

Figure 1: Comparison between EPICG and baselines on lifelong RL benchmarks. X-axis: tasks, Y-axis: reward. CartPole-Goal with $x_{qoal} \sim \mathcal{N}(0, 0.1)$ and $x_{qoal} \sim \mathcal{N}(0, 0.5)$, LunarLander, CartPole-Mass with $\mu_c = 0.5$ and $\mu_c = 1.0$, and Swimmer.

Algorithm 2 Empirical PAC-Baves that Improves Continuously Under Gaussian Prior) (EPICG)

Input: policy dimension d; learning rate β ; update frequency N; failure probability δ ; the number of steps allowed in each task H; prior evolving speed λ

2: Initialize prior policy mean and derivation $\mu_0, \sigma_0 \in$ \mathbb{R}^{d}

Initialize default policy mean and derivation $\mu_0 \leftarrow$

- $\begin{array}{l} \underline{\mu}_0, \, \sigma_0 \leftarrow \underline{\sigma}_0 \, ; \\ 4: \, \, \mathbf{for} \, \, i = 1, 2, 3, \cdots, K, \cdots \infty \, \, \mathbf{do} \end{array}$ Receive a new task $\mathcal{M}_i \sim \mathcal{D}_i$ and store it in memory.
- if $i \mod N = 0$ then 6: Let l = i/N.
- Sample $\{\theta_{l-1,j}\}_{j \in [M]} \sim \mathcal{N}(\mu_{l-1}, \sigma_{l-1}^2)$ by sam-8: ple $\epsilon_j \sim \mathcal{N}(0, I_d)$ Set initial policy, i.e., initialize parameters for neural network $\theta_{l-1,j} \leftarrow \{\mu_{l-1} + \epsilon_j \odot \sigma_{l-1}\}$

Roll out trajectories $\tau_{k,j}$ using $\{\pi_{\theta_{l-1,j}}\}_{j \in [M]}$ 10: and $\{\mathcal{M}_k\}_{k=i-N+1}^i$ and store into $\tau_{l,j}$ # Update default and Prior parameters by using $\tau_{l,i}$

- $\mu_l \leftarrow \mu_{l-1} \beta \nabla_{\mu} \hat{U}(P, \{\mathcal{M}_k\}, \{\theta_{l-1,j}\}_{j \in [M]}; \mu, \sigma)$ 12:
 $$\begin{split} & \sigma_l \leftarrow \sigma_{l-1} - \beta \nabla_{\sigma} \hat{U}(P, \{\mathcal{M}_k\}, \{\theta_{l-1,j}\}_{j \in [M]}; \mu, \sigma) \\ & \mu_l \leftarrow (1-\lambda) \mu_l + \lambda \mu_l; \ \underline{\sigma} \leftarrow (1-\lambda) \underline{\sigma}_l + \lambda \sigma_l \end{split}$$
- 14:Ēmpty Memory by clearing dataset τ_l

16:end if end for

We denote the optimal expected return as $J^* :=$ $\frac{1}{K}\sum_{i=1}^{K} \mathbb{E}_{\{\theta_l\}_{l=0}^{T-1} \sim P^*}[\mathbb{E}_{\{\tau_l\}_{l=1}^{T} \sim \mathscr{T}}[J_{\mathcal{M}_i}(\pi_{\theta})]].$ We provide the sample complexity of EPICG.

Theorem 3.4 (Sample Complexity). Consider the setting of Theorem 3.3. Given a small $\epsilon > 0$, if the number of tasks K satisfies

$$\begin{split} K = & \max\left(\frac{16NH^2\lambda^2r^2}{s_{\min}(1-\alpha)^3(1+\alpha)\epsilon^2}, \left(\frac{16NH^2}{\epsilon^2}\right)^{\frac{1}{1-\gamma}}\right) \\ & + \widetilde{\mathcal{O}}(N\epsilon^{-4}), \end{split}$$

then with high probability,

Training loss $-(-J^*) \leq \mathcal{O}(\epsilon)$,

where $\widetilde{\mathcal{O}}(\cdot)$ suppresses logarithmic dependence.

Proof. The central part of the proof is Theorem 3.3 (detailed proof in Appendix C.2), and the remaining parts are provided in Appendix C.5.

Algorithm 2 learns a general policy distribution. However, if we are interested in a policy for any specific task, we can sample a policy from P and use an appropriate single-task learning method to fine-tune the policy. Hence, every task gets a "customized" policy. We also provide an Algorithm 3 to reflect this in Section 4.3.

Experiments 4

Experimental Setup 4.1

We experiment with common tasks in lifelong-RL benchmarks used in prior works (Mendez et al., 2020; Fu et al., 2022), including HalfCheetah-gravity, HalfCheetah-bodyparts, Hopper-gravity, Hopperbodyparts, Walker-gravity, Walker-bodyparts. To increase the diversity of lifelong environments, we also create several more lifelong environments, Cartpole-GMM, LunarLander-Uniform, Ant-Direction-Uniform, Ant-Forward-Backward-Bernoulli, Swimmer-Uniform, Humanoid-Direction-Uniform. Details about the above environments can be found in Appendix F.1 and in Table 1. In each lifelong environment, the agents are tested across 2,000 or 1,000 tasks. Each environment has a distinct maximum H. As the sequence of 2,000 or 1,000 tasks unfolds, we update the default policy every N tasks. The effectiveness of our approach is



(a) CartP.-Uniform(b) LunarL.-GMM(c) AntD.-Uniform (d) AntF.B.-Bern. (e) Swi.-Uniform (f) Hum.D.-Uniform

Figure 2: Average reward obtained by EPICG-SAC and EPICG in different environments with different lifelong learning settings. *X*-axis: tasks, *Y*-axis: reward.

assessed by how how fast it learns to maximize return as new tasks emerge.

4.2 Effective Lifelong Learning

Figure 1 evaluates EPICG across several control tasks, all of which are lifelong-RL benchmarks used in prior works (Mendez et al., 2020; Fu et al., 2022). This reveals EPICG's noticeable advantage in most scenarios. EPICG consistently outperforms others. We compared EPICG against: 1. Continual Dreamer (Kessler et al., 2023), state-of-the-art lifelong RL method, 2. VBLRL (Fu et al., 2022), Model based Bayesian lifelong RL method; 3. LPG-FTW (Mendez et al., 2020), a lifelong RL method which assumes a factored representations of the policy parameter space; 3. EWC (Kirkpatrick et al., 2017a), which is a single-model lifelong RL algorithm that achieves comparable performance with LPG-FTW as shown in the latter paper; 4. T-HiP-MDP (Killian et al., 2017), which is a model-based lifelong RL baseline; and 5. Single-Task RL. Further details on each baseline can be found in Appendix §B.

4.3 Further Improvement

EPICG effectively learns a shared distribution P of policy parameters for different tasks. Upon receiving new tasks, we learn the policy distribution by using the sampled policy parameter θ . At this point, this approach has already shown effectiveness in our lifelong learning setting. Additionally, we can further improve this θ by optimizing it using data from the new task, customizing it for that particular task. Below we introduce Algorithm 3 (EPICG-SAC), which integrates the EPICG framework with the single task algorithm Soft Actor Critic.

We then compare EPICG-SAC and EPICG for different environments. Figure 2 shows EPICG-SAC achieves faster learning than EPICG.

4.4 Ablation on KL divergence regularization

We first verify the empirical performance when we add the regularizer in (8) compared to having no regularizer by a comparison study. The results are shown in Figure 3, where we observe that adding the regularizer

Algorithm 3 EPICG-SAC

- 1: Input: Same setting as Algorithm 2
- 2: for $i = 1, 2, 3, \dots, K, \dots \infty$ do
- 3: Receive a new task \mathcal{M}_i .
- 4: **if** $i \mod N = 0$ **then**
- 5: Do EPICG Policy Distribution Learning.
- 6: end if
- 7: SAC single-task train-eval loop.
- 8: end for

facilitates fast adaptation, leads to learning a higher reward, and also reduces the variance, which leads to a more stable learning compared to having no regularize.



Figure 3: Comparison of adding $\mathscr{R}(\mathbb{D}_{KL}(P||\bar{P}))$ vs. not. X-axis: tasks, Y-axis: reward.

4.5 Experiments on Memory Size N



Figure 4: Comparison of different update frequency N on (a) CartPole-Uniform; (b) LunarLander-Uniform; (c) Swimmer-Uniform. X-axis: tasks, Y-axis: reward

As we discussed in Theorem 3.3, there is a performance trade-off on the number of tasks N retained in memory. Experimental results have verified this theoretical finding. We can see that in Figure 4 the practical effect of N on the performance of learning is double-sided.

5 Conclusion and Future Works

In this work, we address the challenging problem of lifelong RL and propose a novel algorithm, EPIC(G), for distribution learning and policy sampling. Our approach leverages the concept of a *world policy*, a shared policy distribution across tasks. This world policy is updated continuously, enabling our algorithm to handle both non-stationarity and catastrophic forgetting, achieving best-in-class performance across a suite of complex lifelong RL benchmarks.

Future directions include exploring more accurate ways to obtain the posterior distribution of the policy parameters, as discussed in Section 3.3. Additionally, since the optimization objective in Equation (8) is nonconvex, better performance guarantees could be achieved by investigating multiple optimizations across tasks. Furthermore, constructing deep predictive models for each task to further improve EPIC(G)'s performance.

References

- Abel, D., Barreto, A., Van Roy, B., Precup, D., van Hasselt, H. P., and Singh, S. (2024). A definition of continual reinforcement learning. <u>Advances in</u> <u>Neural Information Processing Systems</u>, 36.
- Abel, D., Jinnai, Y., Guo, S. Y., Konidaris, G., and Littman, M. (2018). Policy and value transfer in lifelong reinforcement learning. In <u>International</u> <u>Conference on Machine Learning</u>, pages 20–29. PMLR.
- Amit, R. and Meir, R. (2018). Meta-learning by adjusting priors based on extended pac-bayes theory. In <u>International Conference on Machine Learning</u>, pages 205–214. PMLR.
- Anand, N. and Precup, D. (2023). Prediction and control in continual reinforcement learning. In <u>Advances in Neural Information Processing Systems</u> (NeurIPS).
- Arnold, S. M., Mahajan, P., Datta, D., Bunner, I., and Zarkias, K. S. (2020). learn2learn: A library for metalearning research. arXiv preprint arXiv:2008.12284.
- Boutilier, C., Cohen, A., Daniely, A., Hassidim, A., Mansour, Y., Meshi, O., Mladenov, M., and Schuurmans, D. (2018). Planning and learning with stochastic action sets. arXiv preprint arXiv:1805.02363.
- Chaloner, K. and Verdinelli, I. (1995). Bayesian experimental design: A review. <u>Statistical science</u>, pages 273–304.
- Chandak, Y., Theocharous, G., Nota, C., and Thomas, P. (2020). Lifelong learning with a changing action set. In <u>Proceedings of the AAAI Conference on</u> Artificial Intelligence, volume 34, pages 3373–3380.

- Dann, C., Mohri, M., Zhang, T., and Zimmert, J. (2021). A provably efficient model-free posterior sampling method for episodic reinforcement learning. <u>Advances in Neural Information Processing Systems</u>, 34:12040–12051.
- Donsker, M. D. and Varadhan, S. S. (1983). Asymptotic evaluation of certain markov process expectations for large time. iv. <u>Communications on pure and applied</u> mathematics, <u>36(2):183–212</u>.
- Dziugaite, G. K., Hsu, K., Gharbieh, W., Arpino, G., and Roy, D. M. (2020). On the role of data in PAC-Bayes bounds. arXiv:2006.10929 [cs, stat].
- Fard, M. and Pineau, J. (2010). PAC-Bayesian Model Selection for Reinforcement Learning. <u>Advances</u> <u>in Neural Information Processing Systems</u>, 23:1624– 1632.
- Fard, M. M., Pineau, J., and Szepesvari, C. (2012). PAC-Bayesian Policy Evaluation for Reinforcement Learning. arXiv:1202.3717 [cs, stat].
- Flynn, H., Reeb, D., Kandemir, M., and Peters, J. (2022). Pac-bayesian lifelong learning for multi-armed bandits. <u>Data Mining and Knowledge Discovery</u>, 36(2):841–876.
- Freedman, D. A. (1975). On tail probabilities for martingales. <u>the Annals of Probability</u>, pages 100–118.
- Fu, H., Yu, S., Littman, M., and Konidaris, G. (2022). Model-based lifelong reinforcement learning with bayesian exploration. <u>Advances in Neural</u> Information Processing Systems, 35:32369–32382.
- Garg, S., Tsipras, D., Liang, P. S., and Valiant, G. (2022). What can transformers learn in-context? a case study of simple function classes. <u>Advances in</u> <u>Neural Information Processing Systems</u>, 35:30583– 30598.
- Germain, P., Lacasse, A., Laviolette, F., and Marchand, M. (2009). PAC-Bayesian learning of linear classifiers. In <u>Proceedings of</u> <u>the 26th Annual International Conference on</u> <u>Machine Learning</u>, ICML '09, pages 353–360, New York, NY, USA. Association for Computing Machinery.
- Ha, D. and Schmidhuber, J. (2018). World models. arXiv preprint arXiv:1803.10122.
- Kessler, S., Ostaszewski, M., Bortkiewicz, M., Żarski, M., Wolczyk, M., Parker-Holder, J., Roberts, S. J., Mi, P., et al. (2023). The effectiveness of world models for continual reinforcement learning. In <u>Conference on Lifelong Learning Agents</u>, pages 184– 204. PMLR.

- Khetarpal, K., Riemer, M., Rish, I., and Precup, D. (2022). Towards continual reinforcement learning: A review and perspectives. <u>Journal of Artificial</u> Intelligence Research, 75:1401–1476.
- Killian, T. W., Daulton, S., Konidaris, G., and Doshi-Velez, F. (2017). Robust and efficient transfer learning with hidden parameter markov decision processes. <u>Advances in neural information processing systems</u>, <u>30</u>.
- Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017a). Overcoming catastrophic forgetting in neural networks. <u>Proceedings of the national academy</u> of sciences, 114(13):3521–3526.
- Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2017b). Overcoming catastrophic forgetting in neural networks. <u>Proceedings of the National Academy of Sciences</u>, 114(13):3521–3526.
- Krähenbühl, P., Doersch, C., Donahue, J., and Darrell, T. (2016). Data-dependent Initializations of Convolutional Neural Networks. <u>arXiv:1511.06856</u> [cs].
- Langford, J. and Shawe-Taylor, J. (2002)PAC-Bayes & amp; margins. In Proceedings International Conference of the 15th on Neural Information Processing Systems, NIPS'02, pages 439–446, Cambridge, MA, USA. MIT Press.
- Lecarpentier, E., Abel, D., Asadi, K., Jinnai, Y., Rachelson, E., and Littman, M. L. (2021). Lipschitz lifelong reinforcement learning. In <u>Proceedings</u> of the AAAI Conference on Artificial Intelligence, volume 35, pages 8270–8278.
- Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.
- Majumdar, A., Farid, A., and Sonar, A. (2021). Pacbayes control: learning policies that provably generalize to novel environments. <u>The International Journal</u> of Robotics Research, 40(2-3):574–593.
- McAllester, D. A. (1999). Some PAC-Bayesian Theorems. Machine Learning, 37(3):355–363.
- Mendez, J., Wang, B., and Eaton, E. (2020). Lifelong policy gradient learning of factored policies for faster training without forgetting. Advances in

Neural Information Processing Systems, 33:14398– 14409.

- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., et al. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540):529–533.
- Naik, D. K. and Mammone, R. J. (1992). Meta-neural networks that learn by learning. In <u>Proceedings</u> <u>1992</u>] IJCNN International Joint Conference on Neural Networks, volume 1, pages 437–442 vol.1.
- Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. (2017). Exploring Generalization in Deep Learning. arXiv:1706.08947 [cs].
- Neyshabur, B., Bhojanapalli, S., and Srebro, N. (2018). A PAC-Bayesian Approach to Spectrally-Normalized Margin Bounds for Neural Networks. arXiv:1707.09564 [cs].
- Salimans, T. and Kingma, D. P. (2016). Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. <u>arXiv:1602.07868</u> [cs].
- Saxe, A. M., McClelland, J. L., and Ganguli, S. (2014). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. <u>arXiv:1312.6120</u> [cond-mat, q-bio, stat].
- Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In <u>International conference on machine</u> learning, pages 1889–1897. PMLR.
- Seeger, M. (2002). PAC-Bayesian Generalisation Error Bounds for Gaussian Process Classification. Journal of Machine Learning Research, 3(Oct):233–269.
- Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. <u>Nature</u>, 529(7587):484– 489.
- Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. arXiv preprint arXiv:1712.01815.
- Steinparz, C. A., Schmied, T., Paischer, F., Dinu, M.-C., Patil, V. P., Bitto-Nemling, A., Eghbal-zadeh, H., and Hochreiter, S. (2022). Reactive exploration to cope with non-stationarity in lifelong reinforcement learning. In <u>Conference on Lifelong Learning Agents</u>, pages 441–469. PMLR.

- Thrun, S. and Pratt, L. (1998). Learning to Learn: Introduction and Overview. In Thrun, S. and Pratt, L., editors, <u>Learning to Learn</u>, pages 3–17. Springer US, Boston, MA.
- Veer, S. and Majumdar, A. (2020). Probably Approximately Correct Vision-Based Planning using Motion Primitives. arXiv:2002.12852 [cs, eess, math].
- Wainwright, M. J. (2019). <u>High-dimensional statistics:</u> <u>A non-asymptotic viewpoint</u>, volume 48. Cambridge <u>university press</u>.
- Yuan, R., Gower, R. M., and Lazaric, A. (2022). A general sample complexity analysis of vanilla policy gradient. In <u>International Conference on Artificial</u> Intelligence and Statistics, pages 3332–3380. PMLR.
- Zhu, Y., Mottaghi, R., Kolve, E., Lim, J. J., Gupta, A., Fei-Fei, L., and Farhadi, A. (2017). Target-driven visual navigation in indoor scenes using deep reinforcement learning. In <u>2017 IEEE international</u> <u>conference on robotics and automation (ICRA)</u>, pages 3357–3364. IEEE.

Supplementary Material

A Broader Impact and Ethical Statement

This paper contributes to the ongoing development of Machine Learning, adhering to standard ethical guidelines in research. Our work, aligning with common advancements in the field, does not present any unique ethical dilemmas or societal consequences that require special emphasis. We recognize the importance of responsible use and development of Machine Learning technologies and their potential impact on society. However, our research does not delve into areas that might raise significant ethical or societal concerns. We commit to ethical research practices and consider the broader implications of our work to be aligned with the typical advancements in Machine Learning.

B Related Works

Lifelong Reinforcement Learning: Lifelong learning has been a crucial area of research in machine learning, where the goal is to develop agents that can continuously adapt to new tasks while retaining knowledge from previous experiences. Early foundational works, such as Naik and Mammone (1992) and Thrun and Pratt (1998), explored the basic principles of lifelong learning, setting the stage for more advanced methods. Subsequent research has focused on mitigating catastrophic forgetting and enhancing data efficiency, which are critical challenges in lifelong learning scenarios. Various approaches have been proposed to improve adaptation in lifelong learning. Saxe et al. (2014); Kirkpatrick et al. (2017b); Krähenbühl et al. (2016); Salimans and Kingma (2016) explored strategies for better initialization in deep networks.

Lifelong RL, as an extension of lifelong learning, naturally aligns with the agent-environment interaction framework, making it ideal for continual learning Khetarpal et al. (2022). Key contributions to this field include Lipschitz Lifelong RL Lecarpentier et al. (2021) and Abel et al. (2018), which emphasize value transfer and initialization to boost learning efficiency. Chandak et al. (2020) tackles the challenge of evolving action sets, Anand and Precup (2023) introduces a dual-component value function approach for balancing long-term stability and short-term adaptability, and Fu et al. (2022) develops a model-based Bayesian framework that enhances both forward and backward transfer by extracting common structures across tasks. Lifelong RL has been further formalized as a framework where agents continuously learn and adapt, moving beyond static solutions Abel et al. (2024).

Recent baseline algorithms for lifelong RL have made significant advancements. Continual Dreamer Kessler et al. (2023) employs ensemble networks and is task-agnostic, leveraging a world model that can generate tasks for improving learning efficiency. VBLRL Fu et al. (2022) is a model-based method that learns a Bayesian posterior distribution shared across tasks to increase sample efficiency in related tasks. LPG-FTW Mendez et al. (2020) is a policy-gradient-based lifelong method that uses data from previously seen tasks to train policy networks, accelerating the learning of new tasks. EWC Kirkpatrick et al. (2017a) is a single-model lifelong RL algorithm that avoids forgetting by imposing a quadratic penalty, pulling weights back towards values important for previously learned tasks. T-HiP-MDP Killian et al. (2017) is a model-based method that models related tasks using low-dimensional latent embeddings and a Bayesian Neural Network, which captures both shared dynamics across tasks and individual task variations.

Our approach introduces a lifelong RL framework integrating PAC-Bayes theory to learn a policy distribution in non-stationary environments, ensuring effective knowledge retention and adaptability across tasks throughout the agent's lifetime.

PAC-Bayes Theory: PAC-Bayes theory McAllester (1999) has been extensively used in supervised and deep learning to study generalization bounds Langford and Shawe-Taylor (2002); Seeger (2002); Germain et al. (2009); Dziugaite et al. (2020); Neyshabur et al. (2018, 2017). In recent years, PAC-Bayes theory has been applied to reinforcement learning (RL) Schulman et al. (2015); Fard and Pineau (2010); Fard et al. (2012); Majumdar et al. (2021); Veer and Majumdar (2020), primarily focused on single-task or offline settings, providing a framework for deriving generalization bounds in dynamic and uncertain environments. Our method offers a novel integration of PAC-Bayes theory into lifelong RL, providing a framework that supports continuous learning and adaptation across varying tasks.

C Detailed Proofs

C.1 Proof of Corollary 3.2

Corollary (Decomposition of Training Error). Assume Assumption 3.1 holds, then it holds true that

$$\frac{1}{K} \sum_{i=1}^{K} \mathbb{E}_{\{\theta_l\}_{l=0}^{T-1} \sim P} [-J_{\mathcal{M}_i}(\pi_{\theta})] = \frac{1}{T} \sum_{l=1}^{T} \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{\theta_{l-1} \sim P_{l-1}} \left[-J_{\mathcal{M}_{l,i}}(\pi_{\theta_{l-1}}) \right]$$

Proof. By Assumption 3.1,

$$P(\theta_{T-1}, \dots, \theta_0)$$

$$=P(\theta_{T-1}|\theta_{T-2}, \dots, \theta_0) \times \dots \times P(\theta_1|\theta_0)P(\theta_0)$$

$$=P(\theta_{T-1}|\theta_{T-2}) \times \dots \times P(\theta_1|\theta_0)P(\theta_0)$$

$$\coloneqq P_{T-1} \times \dots \times P_l \times \dots \times P_0$$
(10)

By tower property, we have

$$\begin{split} \mathbb{E}_{\{\theta_l\}_{l=0}^{T-1} \sim P} \left[-J_{\mathcal{M}_i}(\pi_{\theta}) \right] &= \mathbb{E}_{\{\theta_l \sim P_l\}_{l=0}^{T-1}} \left[-J_{\mathcal{M}_i}(\pi_{\theta}) \right] \\ &= \frac{1}{N} \sum_{i=1}^{N} E_{P_{T-2},...,P_0} E_{\theta_{T-1} \sim P_{T-1} \mid P_{T-2},...,P_0} \left[-J_{\mathcal{M}_{T,i}}(\pi_{\theta_{T-1}}) \right] + \\ &\cdots + \frac{1}{N} \sum_{i=1}^{N} E_{\theta_0 \sim P_0} \left[-J_{\mathcal{M}_{T,i}}(\pi_{\theta_0}) \right] \\ &= \frac{1}{N} \sum_{i=1}^{N} E_{P_{T-2}} E_{\theta_{T-1} \sim P_{T-1} \mid P_{T-2}} \left[-J_{\mathcal{M}_{T,i}}(\pi_{\theta_{T-1}}) \right] + \\ &\cdots + \frac{1}{N} \sum_{i=1}^{N} E_{\theta_0 \sim P_0} \left[-J_{\mathcal{M}_{T,i}}(\pi_{\theta_0}) \right] \\ &= \frac{1}{N} \sum_{i=1}^{N} E_{\theta_{T-1} \sim P_{T-1}} \left[-J_{\mathcal{M}_{T,i}}(\pi_{\theta_{T-1}}) \right] + \\ &\cdots + \frac{1}{N} \sum_{i=1}^{N} E_{\theta_0 \sim P_0} \left[-J_{\mathcal{M}_{T,i}}(\pi_{\theta_0}) \right] \\ &= \frac{1}{T} \sum_{i=1}^{N} E_{\theta_0 \sim P_0} \left[-J_{\mathcal{M}_{T,i}}(\pi_{\theta_0}) \right] \\ &= \frac{1}{T} \sum_{i=1}^{N} E_{\theta_0 \sim P_0} \left[-J_{\mathcal{M}_{T,i}}(\pi_{\theta_0}) \right] \\ &= \frac{1}{T} \sum_{i=1}^{N} \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{\theta_{l-1} \sim P_{l-1}} \left[-J_{\mathcal{M}_{l,i}}(\pi_{\theta_{l-1}}) \right] . \end{split}$$

C.2 Azuma-Hoeffiding or Freedmans inequality for martingale difference sequences for RL

We let the Algorithm (1) experience K tasks, for every N task Algorithm (1) performs lifelong learning, i.e., learns the posterior distribution hyperparameters for policy.

Remember, in Algorithm (1), where the lifelong setting happens with K tasks streaming in, we update the default policy every N tasks and estimate the training cost based on the most recent N tasks. The entire learning process consists of a total of T episodes of updates.

Let the distribution of policy has a parameter mean μ and variance-covariance σ^2 for illustration purposes. For each $l \in [T]$ episode, Algorithm (1) proceeds as:

- 1. Sample $\theta_{l-1} \sim P_{l-1}(\theta; \mu_{l-1}, \sigma_{l-1}^2)$ learnt from the previous episode.
- 2. Using θ_{l-1} and N tasks $\{\mathcal{M}_{i,l}\}_{i\in[N]}$ from current episode to collect data τ_l ;
- 3. Using an optimization algorithm and data from Step 1 to learn the posterior distribution $P_l(\theta; \mu_l, \sigma_l^2)$'s hyper parameters;

For *l*'s episode, for policy π_{θ_l} , which is learned using data roll-out by $\theta_{l-1} \sim P_{l-1}(\theta; \mu_{l-1}, \sigma_{l-1}^2)$, its value function with task $\mathcal{M}_{l,i}$ is defined as the total discounted expected rewards,

$$V_{\mathcal{M}_{l,i}}^{\pi_{\theta_l}}(s_{l,1}) \coloneqq J_{\mathcal{M}_{l,i}}(\pi_{\theta_l}) = \mathbb{E}\left[\sum_{h=1}^{H-1} \gamma^{h-1} r_{l,h} | \pi_{\theta_l}, s_{l,1}, \mathcal{M}_{l,i}\right],\tag{11}$$

from a length of H consecutive sample transitions, $s_{l,1}, a_{l,1}, r_{l,1}, s_{l,2}, a_{l,2}, r_{l,2}, \ldots, s_{l,H} \sim \pi_{\theta_l} \times \mathcal{M}_{l,i}$. Note, θ_l is a function of $\tau_l, \mu_{l-1}, \sigma_{l-1}^2$, which is random, and the randomness is dependent on $\tau_l, \mu_{l-1}, \sigma_{l-1}^2$.

The posterior distribution $P(\theta; \mu, \sigma^2)$ defines a randomized θ . Algorithm (1) draws a θ according to $\theta \sim P(\theta; \mu, \sigma^2)$ at each round of the whole process and applies it to learn the hyperparameter of $P(\theta; \mu, \sigma^2)$ on the next round. For notation-wise, if we do not use subscript l, it means the statement holding for general.

For any θ , let S_T be the difference between the expected and empirical objective value function after the T-th round,

$$S_T \coloneqq \sum_{l=1}^T D_l, \quad l \in [T], \tag{12}$$

where $D_l := \sum_{i=1}^N \mathbb{E}_{\tau_l \sim \theta_{l-1} \times \mathcal{M}_l} \left[V_{\mathcal{M}_{l,i}}^{\pi_{\theta_l}}(s_{l,1}) | \mathcal{F}_{l-1} \right] - \sum_{i=1}^N V_{\mathcal{M}_{l,i}}^{\pi_{\theta_l}}(s_{l,1})$. And the filtration $\mathcal{F}_{l-1} = \sigma(\{\theta_k\}_{k \leq l-2}, \{\mathcal{M}_{i,k-1}\}_{i \in [N], k \leq l-1})$ is the σ -algebra generated by the random variables $\{\theta_k\}_{k \leq l-2}$ and $\{\mathcal{M}_{i,k-1}\}_{i \in [N], k \leq l-1}$.

We first show that using Algorithm (1), after T-th lifelong learning updates, with probability at least $1 - \delta$, for a small $\delta \in (0, 1)$, $S_T = \mathcal{O}(\sqrt{T})$.

Theorem C.1. Let $\{D_l\}_{l \leq T}$, and S_T be defined in Equation 12. For fixed N and H, then with probability at least $1 - \delta$,

$$|S_T| \lesssim \sqrt{\frac{1}{2} \left(\ln \frac{2}{\delta} \right) T N^2 H^2},\tag{13}$$

Furthermore, if $|D_l| \leq b$ for all $l \leq T$, and let

$$\tilde{S}_T = \sum_{l=1}^T \mathbb{E}\left[D_l^2 | \mathcal{F}_{l-1}\right],\tag{14}$$

then with probability at least $1 - \delta$, for $\lambda \in [0, \frac{1}{h}]$,

$$|S_T| \lesssim \frac{1}{\lambda} \ln \frac{2}{\delta} + \lambda \tilde{S}_T \le \frac{1}{\lambda} \ln \frac{2}{\delta} + \lambda T N^2 H^2, \tag{15}$$

Proof. Firstly, the θ_l comes from the posterior distribution $P(\theta; \mu, \sigma^2)$, which depends on $\{\theta_k\}_{k \leq l-1}$ and $\{\mathcal{M}_{i,l}\}_{i \in [N]}$. Furthermore, for a fixed θ_l , we see that D_l is \mathcal{F}_{l-1} measurable. So given that $D_l = \sum_{i=1}^{N} \mathbb{E}\left[V_{\mathcal{M}_{i,l}}^{\pi_{\theta_l}}(s_{l,1}) | \mathcal{F}_{l-1}\right] - \sum_{i=1}^{N} V_{\mathcal{M}_{i,l}}^{\pi_{\theta_l}}(s_{l,1})$, we have $\mathbb{E}\left[D_l | \mathcal{F}_{l-1}\right] = 0$. And $\{D_l\}_{l \in [T]}$ is a martingale difference sequence of functions of θ . Furthermore, $\sum_{i=1}^{N} V_{\mathcal{M}_{i,l}}^{\pi_{\theta_l}}(s_{T,1}) \leq NH$ because of Equation (11) and the reward $r_{l,h}$ is in [0, 1] in Theorem 3.3. And $\{D_l\}_{l \in [T]}$ is a bounded martingale difference sequence. In other words, $D_l \in [a_l, b_l]$, with $a_l = -NH$, $b_l = NH$, and b = NH, this is true because in Equation (11), the reward $r_{l,h}$ is in [0, 1] in Theorem 3.3. Then based on the conclusion of Theorem D.3, the Azuma-Hoeffding Inequality for bounded martingale difference sequence, we get the result in Equation (13).

For Equation (15), we use Theorem D.4, the Freedmans Inequality for martingale difference sequence. For any $\lambda \in [0, \frac{1}{b}]$, where $|D_l| \leq b$, we have

$$\mathbb{P}\left\{S_T \ge t | \mathcal{F}_{T-1}\right\} = \mathbb{P}\left\{e^{\lambda S_T} \ge e^{\lambda t} | \mathcal{F}_{T-1}\right\} \le e^{-\lambda t} \mathbb{E}\left[e^{\lambda S_T} | \mathcal{F}_{T-1}\right] \\
\le e^{-\lambda t} e^{\lambda^2 \tilde{S}_T} = e^{-\lambda t + \lambda^2 \tilde{S}_T}.$$
(16)

Thus,

$$\mathbb{P}\left\{S_T \ge t\right\} \le \mathbb{E}_{\mathcal{F}_{T-1}}\left[e^{-\lambda t + \lambda^2 \tilde{S}_T}\right].$$
(17)

Repeating this argument for $-S_T$, we get the same bound, so overall,

$$\mathbb{P}\left\{|S_T| \ge t\right\} \le 2\mathbb{E}_{\mathcal{F}_{T-1}}\left[e^{-\lambda t + \lambda^2 \tilde{S}_T}\right].$$
(18)

Let $\mathbb{E}_{\mathcal{F}_{T-1}}\left[2e^{-\lambda t+\lambda^2 \tilde{S}_T}\right] = \delta$, we get $t = \frac{1}{\lambda} \ln \frac{2}{\delta} + \lambda \tilde{S}_T$. Therefore, with probability at least $1 - \delta$, for $\lambda \in [0, \frac{1}{\delta}]$,

$$|S_T| \lesssim \frac{1}{\lambda} \ln \frac{2}{\delta} + \lambda \tilde{S}_T \stackrel{(a)}{\leq} \frac{1}{\lambda} \ln \frac{2}{\delta} + \lambda T N^2 H^2, \tag{19}$$

where (a) holds since $D_l \in [a_l, b_l]$ almost surely, the conditioned variable $D_l | \mathcal{F}_{l-1}$ also belongs to this interval almost surely, then we have $\tilde{S}_T \leq \sum_{l=1}^T \frac{(a_l - b_l)^2}{4} \leq TN^2 H^2$ by Lemma D.1

Remark of Theorem C.1. Note, if we minimize the right hand side of Equation (15) with respect to λ , we get the optimal $\lambda = \sqrt{\frac{\ln \frac{2}{\delta}}{TN^2H^2}}$, and $|S_T| \lesssim 2\sqrt{\ln \frac{2}{\delta}TN^2H^2}$. Hence, Equation (15) (derived from Freedmans's Inequality) matches Equation (13) (derived from Azuma-Hoeffding Inequality) up to minor constants and logarithmic factors in the general case, and can be much tighter when the variance $\tilde{S}_T = \sum_{l=1}^T \mathbb{E} \left[D_l^2 |\mathcal{F}_{l-1} \right]$ is small.

C.3 PAC-Bayes Bound

The quantity S_T is of our interest as it is the difference between the expected and empirical objective value after the *T*-th round. In the previous Theorem C.1, we show that S_T is bounded for the sampled θ . Our Algorithm 1 keep updating *P* and sampling θ , rendering $\{P_l\}$ and $\{\theta_l\}$, $l = 0, \ldots, T-1$. To abuse the notation, without further reminder, in the following proof, we use $P(\{\theta_l\})$ and $P(\{\theta_l\})$ to denote the joint posterior and prior, and use θ to denote the set for all θ_l , similarly for the hyperparameter in the distribution. However, since Algorithm (1) draws θ from posterior distribution $P(\theta; \mu, \sigma^2)$, we are interested in the expected value of S_T , which is $\mathbb{E}_P[S_T]$. At the same time, we will relate all possible $P(\theta; \mu, \sigma^2)$ to its corresponding "reference" distribution P, the prior distribution of θ , which is selected before we do step 3. Let $q = (\mu, \sigma)$. In the next Lemma, we will control $\mathbb{E}_P[S_T]$ for any q.

Corollary C.2 (Uniform control of all distributions). Let $\theta \sim P(\theta; q)$ come from a parametrized distribution with the same family as \underline{P} , where \underline{P} is a given prior distribution. Let $\{D_l\}_{l \leq T}$ and S_T follow the same definition in Theorem C.1. For any $\lambda > 0$, let $g(\theta) \coloneqq \lambda S_T(\theta) - \lambda^2 \tilde{S}_T$, where \tilde{S}_T is defined either

$$\tilde{S}_T = \sum_{l=1}^T \frac{(a_l - b_l)^2}{8}, \quad (\text{Match Equation (13)})$$
(20)

or

$$\tilde{S}_T = \sum_{l=1}^T \mathbb{E}\left[D_l^2 | \mathcal{F}_{l-1}\right], \quad (\text{Match Equation (15)})$$
(21)

then with probability at least $1 - \delta$, and for all $P(\theta; q)$,

$$\left|\mathbb{E}_{P}\left[S_{T}(\theta)\right]\right| \leq \frac{\mathbb{D}_{KL}(P\|\underline{P}) + \ln\frac{2}{\delta}}{\lambda} + \lambda \mathbb{E}_{P}\left[\tilde{S}_{T}\right],\tag{22}$$

with $\lambda > 0$ for (20), and $\lambda \in [0, \frac{1}{b}]$ required for (21), where $|D_l| \leq b$ and $|a_l| \leq b, |b_l| \leq b$.

Proof. Based on Theorem D.5, the Donsker–Varadhans Representation formula, we have

$$\mathbb{E}_{P}\left[\left|\lambda S_{T}\right|-\lambda^{2}\tilde{S}_{T}\right] \leq \mathbb{D}_{KL}(P\|\underline{P})+\ln\left(\mathbb{E}_{\underline{P}}\left[e^{|\lambda S_{T}|-\lambda^{2}\tilde{S}_{T}}\right]\right) \\ \stackrel{(a)}{\lesssim} \mathbb{D}_{KL}(P\|\underline{P})+\ln\left(\frac{1}{\delta}\mathbb{E}_{\{D_{l}\}_{l\leq T}}\left[\mathbb{E}_{\underline{P}}\left[e^{|\lambda S_{T}|-\lambda^{2}\tilde{S}_{T}}\right]\right]\right) \quad (\text{with probability at least } 1-\delta) \\ \leq \mathbb{D}_{KL}(P\|\underline{P})+\ln\left(\frac{1}{\delta}\mathbb{E}_{\{D_{l}\}_{l\leq T}}\left[\mathbb{E}_{\underline{P}}\left[e^{\lambda\max\{S_{T},-S_{T}\}-\lambda^{2}\tilde{S}_{T}}\right]\right]\right) \\ \leq \mathbb{D}_{KL}(P\|\underline{P})+\ln\left(\frac{1}{\delta}\mathbb{E}_{\{D_{l}\}_{l\leq T}}\left[\mathbb{E}_{\underline{P}}\left[e^{\lambda S_{T}-\lambda^{2}\tilde{S}_{T}}+e^{\lambda(-S_{T})-\lambda^{2}\tilde{S}_{T}}\right]\right]\right) \\ \leq \mathbb{D}_{KL}(P\|\underline{P})+\ln\left(\frac{1}{\delta}\left(\mathbb{E}_{\underline{P}}\left[\mathbb{E}_{\{D_{l}\}_{l\leq T}}\left[e^{\lambda S_{T}-\lambda^{2}\tilde{S}_{T}}\right]+\mathbb{E}_{\{D_{l}\}_{l\leq T}}\left[e^{\lambda(-S_{T})-\lambda^{2}\tilde{S}_{T}}\right]\right]\right)\right) \\ \stackrel{(b)}{\leq} \mathbb{D}_{KL}(P\|\underline{P})+\ln\frac{2}{\delta}.$$

Where (a) is due to the Lemma D.2, the Markov's Inequality. For the \tilde{S}_k defined in Equation (20), (b) holds because of Equation (42) in Theorem D.3. For the \tilde{S}_k defined in Equation (21), it is based on Theorem D.4. Moving $\lambda \mathbb{E}_P \left[\tilde{S}_T \right]$ to the other side of Equation (23), and divided by λ from both sides,

$$\left|\mathbb{E}_{P}\left[S_{T}(\theta)\right]\right| \leq \frac{\mathbb{D}_{KL}(P\|\underline{P}) + \ln\frac{2}{\delta}}{\lambda} + \lambda \mathbb{E}_{P}\left[\tilde{S}_{T}\right]$$

$$(24)$$

г			
L			
L			
L		. 1	

Corollary C.3 (Proof of Theorem 3.3). If we let \tilde{S}_T follow the definition in 20, for any $\lambda > 0$,

$$|\mathbb{E}_P\left[S_T(\theta)\right]| \le \sqrt{2}NH_{\sqrt{T\left(\mathbb{D}_{KL}(P\|\underline{P}) + \ln\frac{2}{\delta} + \frac{\ln 2}{2\ln c}\left(\frac{\mathbb{D}_{KL}\left(P\|\underline{P}\right)}{\ln\left(\frac{2}{\delta}\right)} + 1\right)\right)},\tag{25}$$

and if we let \tilde{S}_T follow the definition in 21, for any $\lambda \in [0, \frac{1}{b}]$, where $|D_l| \leq b$,

$$\left|\mathbb{E}_{P}\left[S_{T}(\theta)\right]\right| \leq \min\left\{2NH\sqrt{T\left(\mathbb{D}_{KL}(P\|\underline{P}) + \ln\frac{\mathcal{O}(\ln T)}{\delta}\right)}, 2NH\left(\mathbb{D}_{KL}(P\|\underline{P}) + \ln\frac{\mathcal{O}(\ln T)}{\delta}\right)\right\}, \quad (26)$$

Proof. The next step would be to optimize the λ in Equation (22), to get the tightest upper bound. However, the value of λ that minimizes Equation (22) depends on P, whereas we would like to have a result that holds for all possible distributions simultaneously, which is not possible. So we do a discretization of λ . We make a grid of λ 's value in a form of a geometric sequence and for each value of $\mathbb{D}_{KL}(P \parallel P)$, we pick a value of λ from the grid, which is the closest to the one that minimizes the right-hand side of Equation (22) upon to some minor errors. First, in Equation (22), we get

$$\lambda^* = \underset{\lambda}{\arg\min} \frac{\mathbb{D}_{KL}(P \| \underline{P}) + \ln \frac{2}{\delta}}{\lambda} + \lambda \mathbb{E}_P \left[\tilde{S}_T \right] = \sqrt{\frac{\mathbb{D}_{KL}(P \| \underline{P}) + \ln \frac{2}{\delta}}{\mathbb{E}_P \left[\tilde{S}_T \right]}}.$$
(27)

Then putting λ^* back to Equation (22), we get

$$\mathbb{E}_{P}\left[S_{T}(\theta)\right] \leq 2\sqrt{\left(\mathbb{D}_{KL}(P\|\underline{P}) + \ln\frac{2}{\delta}\right)\mathbb{E}_{P}\left[\tilde{S}_{T}\right]}$$
(28)

Moreover, $\mathbb{D}_{KL}(P \| \underline{P}) \ge 0$; note that, if the $\mathbb{D}_{KL}(P \| \underline{P}) = 0$, we get

$$\lambda^{**} = \arg\min_{\lambda} \frac{\ln \frac{2}{\delta}}{\lambda} + \lambda \mathbb{E}_P\left[\tilde{S}_T\right] = \sqrt{\frac{\ln \frac{2}{\delta}}{\mathbb{E}_P\left[\tilde{S}_T\right]}},\tag{29}$$

putting λ^{**} back to Equation (22) with $\mathbb{D}_{KL}(P \| \underline{P}) = 0$, we get

$$|\mathbb{E}_P\left[S_T(\theta)\right]| \le 2\sqrt{\ln\frac{2}{\delta}\mathbb{E}_P\left[\tilde{S}_T\right]}.$$
(30)

Here λ^{**} is also a lower bound for the λ .

Now if we let \tilde{S}_T follow the definition in (21), we have $\mathbb{E}_P\left[\tilde{S}_T\right] \leq Tb^2$ by the definition of \tilde{S}_T in Equation (20) and (21), and $|D_l| \leq b$. Thus, we have $\lambda \in \left[\frac{1}{b}\sqrt{\frac{\ln \frac{2}{\delta}}{T}}, \min\left\{\sqrt{\frac{\mathbb{D}_{KL}(P||\underline{P}) + \ln \frac{2}{\delta}}{\mathbb{E}_P[\tilde{S}_T]}}, \frac{1}{b}\right\}\right]$, note for such \tilde{S}_T , we required $\lambda \leq \frac{1}{b}$.

However, in the setting $\mathbb{D}_{KL}(P||P) > 0$, the value of λ that minimizing right hand side of Equation (22) is given by Equation (27), which depends on P, as early mentioned, thus we use the geometric sequence $\{\lambda_j\}_{j=0}^{J-1}$ over the range $\left[\frac{1}{b}\sqrt{\frac{\ln\frac{2}{\delta}}{T}}, \frac{1}{b}\right]$, for $\lambda_j = c^j \frac{1}{b}\sqrt{\frac{\ln\frac{2}{\delta}}{T}}$, for some c > 1 and $j = 0, \ldots, J-1$. Now we have the geometric series satisfy $c^{J-1}\frac{1}{b}\sqrt{\frac{\ln\frac{2}{\delta}}{T}} \leq \frac{1}{b}$ so as long as $J-1 = \left[\frac{1}{\ln c}\ln\sqrt{\frac{T}{\ln\frac{2}{\delta}}}\right]$.

If
$$\min\left\{\sqrt{\frac{\mathbb{D}_{KL}(P\|\bar{P}) + \ln\frac{2}{\delta}}{\mathbb{E}_{P}[\tilde{S}_{T}]}}, \frac{1}{b}\right\} = \sqrt{\frac{\mathbb{D}_{KL}(P\|\bar{P}) + \ln\frac{2}{\delta}}{\mathbb{E}_{P}[\tilde{S}_{T}]}}$$
, so there are at most total $J = \left\lceil \frac{1}{\ln c} \ln \sqrt{\frac{T}{\ln\frac{2}{\delta}}} \right\rceil + 1 \lambda$'s.

We go back to the proof of Corollary C.2, let $\delta = \sum_{j=0}^{J} \delta_j$, with $\delta_j = \frac{1}{J} \delta$, $j \in \mathbb{N}_{\geq 0}$. Then for any $\delta_j = \frac{1}{J} \delta$, we have

$$\mathbb{P}\left(\mathbb{E}_{P}\left[|\lambda S_{T}| - \lambda^{2} \tilde{S}_{T}\right] \geq \mathbb{D}_{KL}(P \| \underline{P}) + \ln \frac{2}{\delta_{j}} \middle| \lambda \in \left[\frac{1}{b} \sqrt{\frac{\ln \frac{2}{\delta}}{T}}, \frac{1}{b}\right]\right) \\
\stackrel{(a)}{\leq} \mathbb{P}\left(\mathbb{D}_{KL}(P \| \underline{P}) + \ln \left(\mathbb{E}_{\underline{P}}\left[e^{|\lambda S_{T}| - \lambda^{2} \tilde{S}_{T}}\right]\right) \\
\geq \mathbb{D}_{KL}(P \| \underline{P}) + \ln \left(\frac{1}{\delta_{j}} \mathbb{E}_{\{D_{l}\}_{l \leq T}}\left[\mathbb{E}_{\underline{P}}\left[e^{|\lambda S_{T}| - \lambda^{2} \tilde{S}_{T}}\right]\right]\right) \middle| \lambda \in \left[\frac{1}{b} \sqrt{\frac{\ln \frac{2}{\delta}}{T}}, \frac{1}{b}\right]\right),$$
(31)

where (a) holds because $\mathbb{E}_{P}\left[|\lambda S_{T}| - \lambda^{2} \tilde{S}_{T}\right] \leq \mathbb{D}_{KL}(P \| P) + \ln\left(\mathbb{E}_{P}\left[e^{|\lambda S_{T}| - \lambda^{2} \tilde{S}_{T}}\right]\right)$ almost surely, and $\mathbb{D}_{KL}(P \| P) + \ln \frac{2}{\delta_{j}} \geq \mathbb{D}_{KL}(P \| P) + \ln\left(\frac{1}{\delta_{j}}\mathbb{E}_{\{D_{l}\}_{l \leq T}}\left[\mathbb{E}_{P}\left[e^{|\lambda S_{T}| - \lambda^{2} \tilde{S}_{T}}\right]\right]\right)$ almost surely. Here $\left|$ indicates given not conditional on. Following this, we have

$$\begin{split} & \mathbb{P}\left(\mathbb{E}_{P}\left[|\lambda S_{T}|-\lambda^{2}\tilde{S}_{T}\right] \geq \mathbb{D}_{KL}(P\|P) + \ln\frac{2}{\delta_{j}} \middle| \lambda \in \left[\frac{1}{b}\sqrt{\frac{\ln\frac{2}{\delta}}{T}}, \frac{1}{b}\right]\right) \\ \leq & \mathbb{P}\left(\mathbb{D}_{KL}(P\|P) + \ln\left(\mathbb{E}_{P}\left[e^{|\lambda S_{T}|-\lambda^{2}\tilde{S}_{T}}\right]\right) \\ \geq & \mathbb{D}_{KL}(P\|P) + \ln\left(\frac{1}{\delta_{j}}\mathbb{E}_{\{D_{l}\}_{l\leq T}}\left[\mathbb{E}_{P}\left[e^{|\lambda S_{T}|-\lambda^{2}\tilde{S}_{T}}\right]\right]\right) \middle| \lambda \in \left[\frac{1}{b}\sqrt{\frac{\ln\frac{2}{\delta}}{T}}, \frac{1}{b}\right]\right) \\ = & \mathbb{P}\left(\mathbb{E}_{P}\left[e^{|\lambda S_{T}|-\lambda^{2}\tilde{S}_{T}}\right] \geq \frac{1}{\delta_{j}}\mathbb{E}_{\{D_{l}\}_{l\leq T}}\left[\mathbb{E}_{P}\left[e^{|\lambda S_{T}|-\lambda^{2}\tilde{S}_{T}}\right]\right] \middle| \lambda \in \left[\frac{1}{b}\sqrt{\frac{\ln\frac{2}{\delta}}{T}}, \frac{1}{b}\right]\right) \\ \leq & \max_{\lambda \in \left[\frac{1}{b}\sqrt{\frac{\ln\frac{2}{\delta}}{T}}, \frac{1}{b}\right]}\mathbb{P}\left(\mathbb{E}_{P}\left[e^{|\lambda S_{T}|-\lambda^{2}\tilde{S}_{T}}\right] \geq \frac{1}{\delta_{j}}\mathbb{E}_{\{D_{l}\}_{l\leq T}}\left[\mathbb{E}_{P}\left[e^{|\lambda S_{T}|-\lambda^{2}\tilde{S}_{T}}\right]\right]\right) \\ \leq & \sum_{\lambda \in \{\lambda_{1}, \dots, \lambda_{J}\}}\mathbb{P}\left(\mathbb{E}_{P}\left[e^{|\lambda S_{T}|-\lambda^{2}\tilde{S}_{T}}\right] \geq \frac{1}{\delta_{j}}\mathbb{E}_{\{D_{l}\}_{l\leq T}}\left[\mathbb{E}_{P}\left[e^{|\lambda S_{T}|-\lambda^{2}\tilde{S}_{T}}\right]\right]\right) \\ \leq & \sum_{j=1}^{J}\mathbb{P}\left(\mathbb{E}_{P}\left[e^{|\lambda S_{T}|-\lambda^{2}\tilde{S}_{T}}\right] \geq \frac{1}{\delta_{j}}\mathbb{E}_{\{D_{l}\}_{l\leq T}}\left[\mathbb{E}_{P}\left[e^{|\lambda S_{T}|-\lambda^{2}\tilde{S}_{T}}\right]\right]\right) \\ \leq & \sum_{j=1}^{J}\mathbb{P}\left(\mathbb{E}_{P}\left[e^{|\lambda S_{T}|-\lambda^{2}\tilde{S}_{T}}\right] \geq \frac{1}{\delta_{j}}\mathbb{E}_{\{D_{l}\}_{l\leq T}}\left[\mathbb{E}_{P}\left[e^{|\lambda S_{T}|-\lambda^{2}\tilde{S}_{T}}\right]\right]\right) \end{aligned}$$

where (a) we use the discretization of λ , where (b) we apply Lemma D.2.

Thus, we get for all $\lambda \in \left[\frac{1}{b}\sqrt{\frac{\ln\frac{2}{\delta}}{T}}, \min\left\{\sqrt{\frac{\mathbb{D}_{KL}(P\|\underline{P}) + \ln\frac{2}{\delta}}{\mathbb{E}_{P}[\tilde{S}_{T}]}}, \frac{1}{b}\right\}\right]$, we can obtain the inequality as in Equation (28) $|\mathbb{E}_{P}\left[S_{T}(\theta)\right]| \leq 2\sqrt{\mathbb{E}_{P}\left[\tilde{S}_{T}\right]\left(\mathbb{D}_{KL}(P\|\underline{P}) + \ln\frac{2}{\delta_{j}}\right)}$ $\leq 2\sqrt{\mathbb{E}_{P}\left[\tilde{S}_{T}\right]\left(\mathbb{D}_{KL}(P\|\underline{P}) + \ln\frac{2J}{\delta}\right)}$ $= 2NH\sqrt{T\left(\mathbb{D}_{KL}(P\|\underline{P}) + \ln\frac{2J}{\delta}\right)} = 2NH\sqrt{T\left(\mathbb{D}_{KL}(P\|\underline{P}) + \ln\frac{\mathcal{O}(\ln T)}{\delta}\right)},$ (33)

with probability at least $1 - \delta$.

If min $\left\{\sqrt{\frac{\mathbb{D}_{KL}(P\|\underline{P}) + \ln \frac{2}{\delta}}{\mathbb{E}_{P}[\tilde{S}_{T}]}}, \frac{1}{b}\right\} = \frac{1}{b}$, which implies

$$\mathbb{E}_{P}\left[\tilde{S}_{T}\right] \leq b^{2}(\mathbb{D}_{KL}(P\|\underline{P}) + \ln\frac{2}{\delta})$$
(34)

, and for this value of $\lambda = \frac{1}{b}$, we put Equation (34) back to Equation (27), then we get,

$$|\mathbb{E}_{P} [S_{T}(\theta)]| \leq b \mathbb{D}_{KL}(P \| \underline{P}) + b \ln \frac{2}{\delta} + \frac{1}{b} \mathbb{E}_{P} \left[\tilde{S}_{T} \right]$$

$$\leq 2b(\mathbb{D}_{KL}(P \| \underline{P}) + \ln \frac{2}{\delta}) \leq 2NH(\mathbb{D}_{KL}(P \| \underline{P}) + \ln \frac{2}{\delta})$$
(35)

Then under the same argument we did previously for Equation (33), we have

$$|\mathbb{E}_{P}\left[S_{T}(\theta)\right]| \leq 2b(\mathbb{D}_{KL}(P\|\underline{P}) + \ln\frac{2}{\delta_{j}})$$

$$\leq 2NH(\mathbb{D}_{KL}(P\|\underline{P}) + \ln\frac{2J}{\delta}) = 2NH\left(\mathbb{D}_{KL}(P\|\underline{P}) + \ln\frac{\mathcal{O}(\ln T)}{\delta}\right).$$
(36)

Note, the range of λ depends on the \tilde{S}_T , which is sample dependent, thus we have the bound also depends on the sample.

Now if we let \tilde{S}_T follow the definition in (20), now $\mathbb{E}_P\left[\tilde{S}_T\right] = \tilde{S}_T$ since \tilde{S}_T is not random. Now the λ does not have an upper bound. We use the geometric sequence over the range of $\left[\sqrt{\frac{\ln\frac{2}{\delta}}{\tilde{S}_T}},\infty\right]$, where $\sqrt{\frac{\ln\frac{2}{\delta}}{\tilde{S}_T}}$ is given when $\mathbb{D}_{KL}(P\|P) = 0$. We use the same argument, let $\lambda_j = c^j \sqrt{\frac{\ln\frac{2}{\delta}}{\tilde{S}_T}}$ for some c > 1 and $j \ge 0$. For given value of $\mathbb{D}_{KL}(P\|P)$, the optimal λ_j in (22) equals to $\sqrt{\frac{\mathbb{D}_{KL}(P\|P) + \ln\frac{2}{\delta}}{\tilde{S}_T}}$, which requires j is the solution of $c^j \sqrt{\frac{\ln\frac{2}{\delta}}{\tilde{S}_T}} = \sqrt{\frac{\mathbb{D}_{KL}(P\|P) + \ln\frac{2}{\delta}}{\tilde{S}_T}}$, and we floor the value of j to the nearest integer, which is $\left[\ln\left(\frac{\mathbb{D}_{KL}(P\|P)}{\ln(\frac{2}{\delta})} + 1\right)/(2\ln c)\right] \le \left(\ln\left(\frac{\mathbb{D}_{KL}(P\|P)}{\ln(\frac{2}{\delta})}\right) + 1\right)/(2\ln c).$

As the same procedures in Equation (32) we used for deriving Equation (33), we go back to the proof of Corollary C.2, we let $\delta = \sum_{j=0}^{\infty} \delta_j = \sum_{j=0}^{\infty} 2^{-(j+1)} \delta$, with $\delta_j = 2^{-(j+1)} \delta$, $j \in \mathbb{N}_{\geq 0}$.

Then with a similar argument in Equation (32), for any $\delta_j = 2^{-(j+1)}\delta$, we have we have

$$\mathbb{P}\left(\mathbb{E}_{P}\left[|\lambda S_{T}|-\lambda^{2}\tilde{S}_{T}\right] \geq \mathbb{D}_{KL}(P\|\underline{P})+\ln\frac{2}{\delta_{j}}\left|\lambda\in\left[\sqrt{\frac{\ln\frac{2}{\delta}}{\tilde{S}_{T}}},\infty\right]\right)\right) \\ \approx \max_{\lambda\in\{\lambda_{1},\ldots\}}\mathbb{P}\left(\mathbb{E}_{\underline{P}}\left[e^{|\lambda S_{T}|-\lambda^{2}\tilde{S}_{T}}\right]\geq\frac{1}{\delta_{j}}\mathbb{E}_{\{D_{l}\}_{l\leq T}}\left[\mathbb{E}_{\underline{P}}\left[e^{|\lambda S_{T}|-\lambda^{2}\tilde{S}_{T}}\right]\right]\right) \\ \leq \sum_{j=1}^{\infty}\mathbb{P}\left(\mathbb{E}_{\underline{P}}\left[e^{|\lambda_{j}S_{T}|-\lambda_{j}^{2}\tilde{S}_{T}}\right]\geq\frac{1}{\delta_{j}}\mathbb{E}_{\{D_{l}\}_{l\leq T}}\left[\mathbb{E}_{\underline{P}}\left[e^{|\lambda_{j}S_{T}|-\lambda_{j}^{2}\tilde{S}_{T}}\right]\right]\right) \\ \leq \sum_{j=1}^{\infty}\times\delta_{j}=\sum_{j=1}^{\infty}\delta2^{-(j+1)}=\delta.$$
(37)

In the end, we get for all $\lambda \in \left[\sqrt{\frac{\ln \frac{2}{\delta}}{\tilde{S}_T}}, \infty\right]$, we can obtain the inequality as in Equation (28)

$$\begin{aligned} |\mathbb{E}_{P}\left[S_{T}(\theta)\right]| &\leq 2\sqrt{\tilde{S}_{T}\left(\mathbb{D}_{KL}(P\|\underline{P}) + \ln\frac{2}{\delta_{j}}\right)} \\ &\leq 2\sqrt{\tilde{S}_{T}\left(\mathbb{D}_{KL}(P\|\underline{P}) + \ln\frac{2*2^{j}}{\delta}\right)} \\ &\leq 2\sqrt{\sum_{l=1}^{T}\frac{(a_{l} - b_{l})^{2}}{8}\left(\mathbb{D}_{KL}(P\|\underline{P}) + \ln\frac{2}{\delta} + \frac{\ln 2}{2\ln c}\left(\ln\left(\frac{\mathbb{D}_{KL}(P\|\underline{P})}{\ln\left(\frac{2}{\delta}\right)}\right) + 1\right)\right)} \\ &\leq \sqrt{2}NH\sqrt{T\left(\mathbb{D}_{KL}(P\|\underline{P}) + \ln\frac{2}{\delta} + \frac{\ln 2}{2\ln c}\left(\ln\left(\frac{\mathbb{D}_{KL}(P\|\underline{P})}{\ln\left(\frac{2}{\delta}\right)}\right) + 1\right)\right)}. \end{aligned}$$
(38)

By utilizing Equation (10), the joint distribution can be decomposed into products of independent distributions that are solely dependent on the preceding episode, which can be successively absorbed into the filtration we

defined. By $P = \prod_{l=0}^{T-1} P_l$ and $\underline{P} = \prod_{l=0}^{T-1} \underline{P}_l$, we have

$$\mathbb{D}_{KL}(P\|\underline{P}) = \sum_{l=1}^{T} \mathbb{D}_{KL}(P_{l-1}\|\underline{P}_{l-1}), \qquad (39)$$

putting Equation (39) back into (38), we have

$$|\mathbb{E}_{P}\left[S_{T}(\theta)\right]| \leq \sqrt{2}NH \sqrt{T\left(\sum_{l=1}^{T} \mathbb{D}_{KL}\left(P_{l-1} \| \underline{P}_{l-1}\right) + \ln\frac{2}{\delta} + \frac{\ln 2}{2\ln c} \left(\ln\left(\frac{\sum_{l=1}^{T} \mathbb{D}_{KL}\left(P_{l-1} \| \underline{P}_{l-1}\right)}{\ln(\frac{2}{\delta})}\right) + 1\right)\right)}.$$
 (40)

Next, by Lemma C.4, we have

$$\sum_{l=1}^{T-1} \mathbb{D}_{KL} \left(P_l \| \underline{P}_l \right) \le \frac{2\lambda^2 r^2}{s_{\min}(1-\alpha)^2} \frac{1-\alpha^{2(T-1)}}{1-\alpha^2}$$

Return to T := K/N, then take $\delta = 2 \exp(-K)$, choose c such that $\frac{\ln 2}{2 \ln c} = 1$, and by the inequality $\sqrt{a+b} \le \sqrt{a} + \sqrt{b}$ and $\ln(K) + K \le \sqrt{2}K$ for any K > 0, some basic algebra, we get the final bound in Theorem 3.3 Equation (5).

Lemma C.4. Suppose Assumption 3.1 holds. Then, for any $l \in \{1, ..., T-1\}$, the following bound on the KL divergence holds:

$$\sum_{l=1}^{T-1} \mathbb{D}_{KL}(P_l \| \underline{P}_l) \le \frac{2\lambda^2 r^2}{s_{\min} (1-\alpha)^2} \cdot \frac{1-\alpha^{2(T-1)}}{1-\alpha^2}$$

Proof. For any l, we denote lth λ as λ_l , so $\lambda_1 = \lambda$, $\lambda_l = \alpha^{l-1} \lambda$. First we have

$$\mathbb{D}_{KL}\left(P_{l}\|\underline{P}_{l}\right) \leq \frac{2\|P_{l}-\underline{P}_{l}\|_{\infty}^{2}}{s_{min}}$$

Further, given the updating rule, $\underline{P}_{l} = \lambda_{l} \underline{P}_{l-1} + (1 - \lambda_{l}) P_{l}$, we have

$$\begin{split} \|P_{l} - P_{l}\|_{\infty} &= \|P_{l} - \lambda_{l}P_{l-1} - (1 - \lambda_{l})P_{l}\|_{\infty} = \|\lambda_{l}P_{l} - \lambda_{l}P_{l-1}\|_{\infty} \\ &= \|\lambda_{l}P_{l} - \lambda_{l}(\lambda_{l-1}P_{l-2} + (1 - \lambda_{l-1})P_{l-1})\|_{\infty} = \|\lambda_{l}(P_{l} - P_{l-1}) + \lambda_{l}\lambda_{l-1}(P_{l-1} - P_{l-2})\|_{\infty} \\ &= \|\lambda_{l}(P_{l} - P_{l-1}) + \lambda_{l}\lambda_{l-1}(P_{l-1} - (\lambda_{l-2}P_{l-3} + (1 - \lambda_{l-2})P_{l-2}))\|_{\infty} \\ &= \|\lambda_{l}(P_{l} - P_{l-1}) + \lambda_{l}\lambda_{l-1}(P_{l-1} - P_{l-2}) + \lambda_{l}\lambda_{l-1}\lambda_{l-2}(P_{l-2} - P_{l-3})\|_{\infty} \\ &= \cdots \\ &= \|\lambda_{l}(P_{l} - P_{l-1}) + \lambda_{l}\lambda_{l-1}(P_{l-1} - P_{l-2}) + \cdots + \lambda_{l}\lambda_{l-1} \cdots \lambda_{2}(P_{2} - P_{1}) + \lambda_{l}\lambda_{l-1}\lambda_{l-2} \cdots \lambda_{1}(P_{1} - P_{0})\|_{\infty} \\ &= \|\alpha^{l-1}\lambda_{1}(P_{l} - P_{l-1}) + \alpha^{l-1+l-2}\lambda_{1}(P_{l-1} - P_{l-2}) + \cdots + \alpha^{l-1+l-2+\dots+1}\lambda_{1}(P_{1} - P_{0})\|_{\infty} \\ &\leq \lambda_{1}r\frac{\alpha^{l-1}}{1 - \alpha}, \end{split}$$

where we use, $\underline{P}_0 = P_0$, $\lambda_l = \alpha \lambda_{l-1}$, and $\|P_l - P_{l-1}\|_{\infty} \leq r$, we have $\mathbb{D}_{KL}(P_l \|\underline{P}_l) \leq \frac{2\lambda^2 r^2 \alpha^{2l-2}}{s_{\min}(1-\alpha)^2}$, thus we have $\sum_{l=1}^{T-1} \mathbb{D}_{KL}(P_l \|\underline{P}_l) \leq \sum_{l=1}^{T-1} \frac{2\lambda^2 r^2 \alpha^{2l-2}}{s_{\min}(1-\alpha)^2} \leq \frac{2\lambda^2 r^2}{s_{\min}(1-\alpha)^2} \frac{1-\alpha^{2(T-1)}}{1-\alpha^2}$.

C.4 Proof of Theorem 3.4

Lemma C.5 (Sample Complexity For Policy Gradient). Consider the setting of Thm. 3.3. Given a small $\epsilon > 0$, with proper choice of learning rate β , If the number of iterations T satisfies $T = \widetilde{\mathcal{O}}(\epsilon^{-4})$. Then Expected loss $-(-J^*) \leq \mathcal{O}(\epsilon)$.

Proof. Refer the Corollary C.1 in Yuan et al. (2022) for details.

We then begin to prove Theorem 3.4.

Proof. By the proof of Theorem 3.3, we have

 $|\text{Expected loss} - \text{Training error}| \leq \mathscr{R}(\mathbb{D}_{KL}(P||\underline{P})), \text{ then we impose the following conditions:}$

- 1. let $\mathscr{R}(\mathbb{D}_{KL}(P \| \underline{P})) \leq \frac{\epsilon}{2}$,
- 2. and let Expected loss $-(-J^*) \leq \mathcal{O}(\frac{\epsilon}{2})$.

By satisfying both conditions 1 and 2, we obtain, Training error $-(-J^*) \leq \mathcal{O}(\epsilon)$. The value of K can then be determined to satisfy these conditions.

D Auxiliary Theorems and Lemmas

Lemma D.1 (Popoviciu's inequality on variances, Wainwright (2019)). For bounded random variable $x \in [a, b]$, then $\operatorname{Var}[x] \leq \frac{(b-a)^2}{4}$

Lemma D.2 (Markov's Inequality, Wainwright (2019)). For any non-negative random variable x, it holds that $\mathbb{P}(x \ge t) \le \frac{\mathbb{E}[x]}{t}$. Taking $t = \frac{\mathbb{E}[x]}{\delta}$, where $\delta \in (0, 1)$, it results in with probability at least $1 - \delta$, $0 \le x \le \frac{\mathbb{E}[x]}{\delta}$.

Theorem D.3 (Azuma-Hoeffding Inequality, Wainwright (2019)). For a sequence of Martingale Difference Sequence random variable $\{D_l\}_{l=1}^T$, if we have $D_l \in [a_l, b_l]$ almost sure for some constant $[a_l, b_l]$ and l = 1, 2, ..., T, the summation $S_T := \sum_{l=1}^T D_l$, and let $\tilde{S}_T = \frac{\sum_{l=1}^T (b_l - a_l)^2}{8}$ then:

$$\mathbb{P}\left(|S_T| \ge t\right) \le 2e^{\frac{-t^2}{S_T}} \tag{41}$$

Equivalently, the moment-generating function satisfies

$$\mathbb{E}\left[e^{\lambda S_T}\right] \le e^{\lambda^2 \tilde{S}_T}.\tag{42}$$

Furthermore, if we choose $t = \sqrt{\frac{1}{2} \ln \frac{2}{\delta} \sum_{l=1}^{T} (b_l - a_l)^2}$, we get $\mathbb{P}\left(|S_T| > \sqrt{\frac{1}{2} \ln \frac{2}{\delta} \sum_{l=1}^{T} (b_l - a_l)^2}\right) \le \delta$.

Theorem D.4 (Freedman's inequality, Freedman (1975)). Let \mathcal{F}_T , $\{D_l\}_{l \leq T}$, follow the definition in Equation (12), and let $|D_l| \leq b$ with probability at least 1 and $\mathbb{E}[D_l|\mathcal{F}_{l-1}] = 0$. Let $S_T := \sum_{l=1}^T D_l$ and let $\tilde{S}_T := \sum_{l=1}^T \mathbb{E}[D_l^2|\mathcal{F}_{l-1}]$. Then for any $\lambda \in [0, \frac{1}{b}]$

$$\mathbb{E}_{\{D_l\}_{l\leq T}}\left[e^{\lambda S_T - \lambda^2 \tilde{S}_T}\right] \leq 1 \tag{43}$$

Proof.

$$\mathbb{E}_{D_T} \left[e^{\lambda D_T} | \mathcal{F}_{T-1} \right] \stackrel{(a)}{\leq} \mathbb{E}_{D_T} \left[1 + \lambda D_T + \lambda^2 D_T^2 | \mathcal{F}_{T-1} \right] = 1 + \lambda^2 \mathbb{E}_{D_T} \left[D_T^2 | \mathcal{F}_{T-1} \right] \stackrel{(b)}{\leq} e^{\lambda^2 \mathbb{E}_{D_T} \left[D_T^2 | \mathcal{F}_{T-1} \right]}$$
(44)

Where (a) holds since $e^x \leq 1 + x + x^2$ for $0 < x \leq 1$, thus, we require $\lambda D_T \leq 1$, so $\lambda \leq \frac{1}{b}$, and (b) holds by $1 + x \leq e^x$. Now we have

$$\mathbb{E}_{\{D_l\}_{l\leq T}} \left[e^{\lambda S_T - \lambda^2 \tilde{S}_T} \right] \stackrel{(a)}{=} \mathbb{E}_{\{D_l\}_{l\leq T}} \left[e^{\lambda S_{T-1} - \lambda^2 \tilde{S}_{T-1} + \lambda D_T - \lambda^2 \mathbb{E}[(D_T)^2 | \mathcal{F}_{T-1}]} \right] \\
\stackrel{(b)}{=} \mathbb{E}_{\{D_l\}_{l\leq T-1}} \left[e^{\lambda S_{T-1} - \lambda^2 \tilde{S}_{T-1}} \times \mathbb{E}_{D_T} \left[e^{\lambda D_T} | \mathcal{F}_{T-1} \right] \times e^{-\lambda^2 \mathbb{E}[(D_T)^2 | \mathcal{F}_{T-1}]} \right] \\
\stackrel{(c)}{\leq} \mathbb{E}_{\{D_l\}_{l\leq T-1}} \left[e^{\lambda S_{T-1} - \lambda^2 \tilde{S}_{T-1}} \right] \\
\stackrel{(45)}{\leq} \dots \\
\stackrel{(45)}{=} \dots$$

where (a) holds by the definition of \tilde{S}_T , and (b) holds by the definition of the definition $D_T | \mathcal{F}_{T-1}$, where (c) holds by (44), and in the last step above we have recursively applied the above argument.

Theorem D.5 (Donsker–Varadhan's Representation formula, Donsker and Varadhan (1983)). Given a probability space $(\mathcal{X}, \mathcal{B})$ and a bounded real-valued function f, where f(x) is a measurable function $f : \mathcal{X} \to \mathbb{R}$, x is a random variable, and any two probability distributions P_0 and P over \mathcal{X} (or, if \mathcal{X} is uncountably infinite, two probability density functions),

$$\mathbb{D}_{KL}\left(P\|\underline{P}\right) \ge \mathbb{E}_{P}\left[f(x)\right] - \ln \mathbb{E}_{\underline{P}}\left[e^{f(x)}\right].$$
(46)

The $\ln \mathbb{E}_{P}\left[e^{f(x)}\right]$ on the right-hand side is the cumulant generating function.

E Policy Function Parameter θ With A Gaussian Prior

E.1 Neural Network Parametrization

In Equation (7), the parameter θ can represent the weights of a neural network. Here, we provide details on how we set up the parameter updates for the neural network weights. Let $\theta = (w_r, b_r)$ denote the random weights and biases of the *r*-th ($r \in \mathbb{N}_{\geq 1}$) network layer. Additionally, let ϵ_r and ϵ_{b_r} be multivariate standard normal distributed random variables. The random weights w_r and biases b_r are defined as follows:

$$w_r = \mu_r \odot (1 + \gamma_r \epsilon_r), \gamma_r = \ln(1 + \exp(\delta_r)), \tag{47}$$

$$b_r = \mu_{b_r} \odot (1 + \gamma_{b_r} \epsilon_{b_r}), \gamma_{b_r} = \ln(1 + \exp(\delta_{b_r})).$$

$$\tag{48}$$

This implies that w_r and b_r are multivariate normal distributed according to:

$$w_r \sim \mathcal{N}(\mu_r, \gamma_r^2 \operatorname{diag}(\mu_r^2))), \quad b_r \sim \mathcal{N}(\mu_{b_r}, \gamma_{b_r}^2 \operatorname{diag}(\mu_{b_r}^2)).$$
 (49)

During optimization in each iteration, a sample of w_r and b_r is drawn from the random network parameters to perform gradient descent.

The indirect sampling according to Equations (47) and (48) ensures that the parameters $\mu_r, \gamma_r, \mu_{b_r}, \gamma_{b_r}$ can be updated. The normal prior $p(\theta)$ is defined as:

$$w_{\underline{r}} \sim \mathcal{N}(\mu_r, \gamma_r^2), \quad b_{\underline{r}} \sim \mathcal{N}(\mu_{b_r}, \gamma_{b_r}^2).$$
 (50)

Thus, the posterior distribution for the neural network is given as $p(\theta|D) = p(\theta)p(D|\theta) / \int \underline{p}(\theta)p(D|\theta)d\theta$, where $p(D|\theta) \coloneqq p(g(\tau)|\theta)$ is the data likelihood. The exact likelihood function $p(D|\theta)$ and posterior policy $p(\theta|D)$ are left as future research, as mentioned in section 3.3 "Posterior Distribution and Prior Distribution".

Instead of analytically deriving $p(\theta|D)$, we assume it belongs to a common distribution family of the prior, but with unknown parameters, which are updated by minimizing the upper bound. Therefore, we approximate the posterior $p(\theta|D)$ by a proposed distribution $q(\theta)$.

Following this approach, we can approximate the posterior $p(\theta|D)$ by updating the parameters of $q(\theta)$ using the indirect sampling chain rule. We first sample a $\theta \sim (\mathcal{N}_{w_r}, \mathcal{N}_{b_r})$, then evaluate the right-hand side in Equation (8).

We can calculate the derivatives of $\hat{U}(P, \{\mathcal{M}_i\}_{i \in [N]}, \{\theta_{l-1,j}\}_{j \in [M]}; \mu, \sigma, \underline{\mu}, \underline{\sigma})$ with respect to $\mu_r, \mu_{b_r}, \delta_r, \delta_{b_r}$ and $\mu_r, \mu_{b_r}, \underline{\delta}_r, \underline{\delta}_{b_r}$ in Equations (47) and (48), which we used in our implementation.¹

F Environment and Experiment

	cart mass	$0.15[\mathcal{N}(1,0.1^2) + 0.15\mathcal{N}(5,0.1^2)] + 0.18[\mathcal{N}(2,0.1^2) + 0.15[\mathcal{N}(2,0.1^2)] + 0$
CartPole-GMM	nola mass	$0.18\mathcal{N}(4,0.1^2)] + 0.34\mathcal{N}(3,0.1^2)$ 0.15[$\mathcal{N}(0,4,0,01^2) + \mathcal{N}(0,5,0,01^2)$]
	pole mass	$0.15[\mathcal{N}(0.2, 0.01^2) + \mathcal{N}(0.3, 0.01^2)] + 0.34\mathcal{N}(0.1, 0.01^2)$
	pole length	$0.15[\mathcal{N}(0.3, 0.01^2) + \mathcal{N}(0.7, 0.01^2)] +$
		$0.18[\mathcal{N}(0.4, 0.01^2) + \mathcal{N}(0.6, 0.01^2)] + 0.34\mathcal{N}(0.5, 0.01^2)$
CartPole-Uniform	cart mass	$\mathcal{U}(1,5)$
	pole mass	$\mathcal{U}(0.1, 0.5)$
	pole length	$\mathcal{U}(0.3, 0.7)$
	main engine power	$0.15[\mathcal{N}(11, 0.1^2) + 0.18\mathcal{N}(12, 0.1^2)] +$
LunarLander-GMM		$0.34[\mathcal{N}(13,0.1^2) + 0.18\mathcal{N}(14,0.1^2)] + 0.15\mathcal{N}(15,0.1^2)$
	side engine power	$0.15[\mathcal{N}(0.45, 0.01^2) + 0.18\mathcal{N}(0.55, 0.01^2)] +$
		$0.34[\mathcal{N}(0.65, 0.01^2) + 0.18\mathcal{N}(0.75, 0.01^2)] +$
		$0.15\mathcal{N}(0.85, 0.01^2)$
LunarLander-Uniform	main engine power	$\mathcal{U}(3,20)$
	side engine power	$\mathcal{U}(0.15, 0.95)$
Swimmer-Uniform	movement direction	$ heta \sim \mathcal{U}(0,\pi)$
${\rm Humanoid}\text{-}{\rm Direction}\text{-}{\rm Uniform}$	movement direction	$\theta \sim \mathcal{U}(0, 2\pi)$
Ant-Direction-Uniform	goal direction	$ heta \sim \mathcal{U}(0, 2\pi)$
Ant-Forward-Backward-Bernoulli	movement direction	$\theta \sim \text{Categorical}(0, \pi; 0.5)$
HalfCheetah-gravity	See Mendez et al. (2020); Fu et al. (2022)	
HalfCheetah-bodyparts	See Mendez et al. (2020); Fu et al. (2022)	
Hopper-gravity	See Mendez et al. (2020); Fu et al. (2022)	
Hopper-bodyparts	See Mendez et al. (2020); Fu et al. (2022)	
Walker-gravity	See Mendez et al. (2020); Fu et al. (2022)	
Hopper-bodyparts	See Mendez et al. (2020); Fu et al. (2022)	

Table 1:	: Different	Lifelong	Environments
----------	-------------	----------	--------------

F.1 OpenAI and MAMuJoCo Environment

F.1.1 CartPole-GMM Environment

In the classic CartPole environment depicted in §F.1 Figure 5a, a pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force of +1 or -1 to the cart. A reward of +1 is provided for every timestep that the pole remains upright. The episode ends when the pole falls or the cart moves far away from the center. Important parameters in the Cartpole Environment are subject to Gaussian Mixture probability distributions to create a diverse of tasks, as in Table 1.

F.1.2 CartPole-Uniform Environment

Similar to the CartPole-GMM, the parameters in the Cartpole Environment are subject to some uniform distributions herein, as in Table 1.

 $^{^{1}}$ We have included the pseudo-code in the Algorithm section. As for the source code, we will publish it on GitHub if our paper is accepted. Thank you.



Figure 5: The illustration of environments. (a) CartPole, (b) LunarLander, (c) Swimmer, (d) Ant, (e) Humanoid, (f) Cheetah, (g) Hopper, (h) Walker

F.1.3 LunarLander-GMM Environment

The general task in LunarLander environment (as shown in §F.1 Figure 5b) is to let lunar module land at a pre-defined goal location. The parameters in the LunarLander Environment are subject to Gaussian Mixture probability distributions herein, as in Table 1.

F.1.4 LunarLander-GMM Environment

The general task in LunarLander environment (as shown in $\SF.1$ Figure 5b) is to let lunar module land at a pre-defined goal location. The parameters in the LunarLander Environment are subject to Gaussian Mixture probability distributions herein, as in Table 1.

F.1.5 LunarLander-Uniform Environment

Similar to the LunarLander-GMM environment, the parameters are subject to uniform probability distributions herein, as in Table 1.

F.1.6 Swimmer-Uniform Environment

The original Swimmer environment (as shown in Figure 5c) is part of the Mujoco simulation suite. The Swimmer is a multi-link agent suspended in a two-dimensional pool, consisting of three or more segments (links) connected by articulation joints (rotors). The goal of the agent is to move as fast as possible towards the right by applying torques to the rotors. The forward movement is measured as the change in the x-coordinate of the swimmer's front tip, while the control cost penalizes the swimmer for large actions. The total reward is the sum of the forward reward and control penalty, and the environment ends when the episode reaches a maximum length of 1000 timesteps.

In the default setup, the swimmer consists of three links, and the agent's observations include the angles and angular velocities of the rotors, as well as the linear velocity of the front tip along the x- and y-axes. The challenge in this environment comes from the need to balance efficient forward movement while minimizing control costs, with the goal being to achieve the highest reward by maximizing forward movement while keeping control actions minimal.

In the modified Swimmer-Uniform environment, the objective has been extended to include movement in any direction, rather than just forward. The direction of movement is determined by randomly sampling an angle θ uniformly from the interval $[0, \pi]$. This angle defines the goal movement direction using a unit vector $[\cos(\theta), \sin(\theta)]$, allowing the swimmer to move in a random direction within the 2D plane.

During each timestep, the agent's forward reward is computed as the dot product of the swimmer's displacement vector (posafter – posbefore) and the goal direction vector. This encourages the swimmer to move in the assigned random direction. As in the original environment, there is a control cost that penalizes excessive torque applied to the rotors, calculated as $0.5 \times 10^{-4} \times \sum \text{action}^2$, where the sum is over the torques applied between the links. The total reward is the sum of the forward reward and the control cost penalty.

The agent's state observation remains the same, consisting of the angles and angular velocities of the rotors, as well as the velocity of the front tip along the x- and y-axes. The episode ends after a maximum length of 1000

timesteps, or when the agent reaches a specified goal. This modification introduces task diversity by randomizing the goal direction, making it a suitable environment for evaluating lifelong learning systems.

F.1.7 Ant-Direction-Uniform Environment

We adopt Ant Direction, Ant Forward-Backward, and Humanoid Direction environments, as shown in Figure 5d, and 5e, whose implementation is from learn2learn Arnold et al. (2020) https://github.com/learnables/learn2learn/tree/master/learn2learn/gym/envs/mujoco.

For Ant-Direction environment (as shown in Figure 5d), the agent controls an ant-like robot in the Mujoco simulator with the objective of moving in a randomly sampled direction on the XY plane. At the start of each task, a random direction is determined by drawing a uniform random variable θ from the interval $[0, 2\pi]$. This angle θ defines the target movement direction as a unit vector $[\sin(\theta), \cos(\theta)]$.

In the Ant-Direction-Uniform implementation, key parameters such as the goal direction are generated randomly for each task by sampling the angle from a uniform distribution. During each timestep, the agent receives a reward based on its forward movement in the direction defined by the goal vector, calculated as the dot product of the goal direction vector and the change in the ant's torso position before and after applying the action, normalized by the timestep dt.

Additional costs are associated with controlling the ant, including a control cost computed as $0.5 \times \sum \arctan^2$, where the sum is over the joint torques applied by the agent. Similarly, a contact cost penalizes the external contact forces experienced by the ant, computed as $0.5 \times 10^{-3} \times \sum \text{contact forces}^2$. The agent also receives a constant survival reward of 1.0 per timestep, incentivizing it to remain upright.

The termination condition for the episode occurs if the ant's torso falls below a height of 0.2 or rises above 1.0, indicating failure or unrealistic behavior. The agent's state observation consists of the joint positions (excluding the torso orientation), joint velocities, and external contact forces, all of which are concatenated into a single vector.

F.1.8 Ant-Forward-Backward-Bernoulli Environment

The original Ant-Forward-Backward environment requires the agent to control an ant-like robot in the Mujoco simulator, with the goal of moving either forward or backward along the X-axis. At the start of each task, the movement direction is randomly sampled from a Bernoulli distribution, where +1 indicates forward movement and -1 indicates backward movement, each with a probability of 0.5. The reward structure in this environment is based on the ant's velocity in the chosen direction, with additional control and contact costs penalizing excessive actions or external forces. The goal is to maximize movement in the assigned direction while minimizing the costs. The episode terminates either when the agent fails to stay upright or when a maximum number of timesteps is reached.

In the modified Ant-Forward-Backward-Bernoulli environment, the movement direction is also determined using a Bernoulli distribution, but the direction is represented by an angle θ , sampled from the set $\{0, \pi\}$, corresponding to forward and backward movement respectively. The goal direction is then defined as a unit vector $[\sin(\theta), \cos(\theta)]$, where $\theta = 0$ indicates forward movement and $\theta = \pi$ indicates backward movement.

During each timestep, the agent's forward reward is computed as the dot product of the ant's displacement vector (pos_after - pos_before) and the goal direction vector. This reward structure encourages efficient movement in the assigned direction. The forward reward is calculated as:

$$reward_fwd = \sum (goal_direction \cdot (pos_after - pos_before))/dt$$

As in the original environment, there is a control cost for large actions, computed as $0.5 \times \sum \text{action}^2$, and a contact cost for external forces, computed as $0.5 \times 10^{-3} \times \sum \text{external forces}^2$. The agent also receives a survival reward of 1.0 per timestep, incentivizing it to remain upright.

The episode terminates if the ant's torso height falls below 0.2 or rises above 1.0, indicating failure or unrealistic behavior. The state observation consists of joint positions (excluding the torso orientation), joint velocities, and external contact forces, all concatenated into a single observation vector. This modified environment introduces variability by randomly assigning forward or backward movement based on a Bernoulli distribution.

F.1.9 Humanoid-Direction-Uniform Environment

The original Humanoid-Direction environment (as shown in Figure 5e) requires the agent to control a humanoid robot in the Mujoco simulator with the objective of moving in a randomly assigned direction in the XY plane. At the start of each task, the direction is randomly sampled from a 2D normal distribution and normalized to represent a unit vector. The agent's goal is to move in this direction as efficiently as possible. The reward function is based on the humanoid's velocity in the assigned direction, with a control cost penalizing excessive actions and an impact cost penalizing large contact forces. A constant alive bonus is added at each timestep to encourage the humanoid to stay upright and moving.

In the original environment, the agent's state observations include the joint positions and velocities, along with external forces acting on the humanoid's body. The agent must maximize its forward velocity while minimizing control and impact costs. The environment ends when the episode reaches its maximum length or the humanoid falls down, indicated by its torso height moving outside a predefined range.

In the modified Humanoid-Direction-Uniform environment, the movement direction is determined by sampling a uniform random angle θ from the interval $[0, 2\pi]$. This angle defines the goal movement direction using the unit vector $[\sin(\theta), \cos(\theta)]$, allowing for movement in any direction on the XY plane.

During each timestep, the agent receives a reward based on its movement in the direction of the goal vector, computed as the dot product of the humanoid's displacement and the goal direction vector. The forward velocity reward is given by $1.25 \times \sum$ goal_direction \cdot (pos_after – pos_before)/dt. Additional costs include a control cost of $0.1 \times \sum$ action², penalizing large actions, and an impact cost of $0.5 \times 10^{-6} \times \sum$ external_forces², penalizing large external forces acting on the humanoid. The total reward is the sum of the forward velocity reward, alive bonus of 5.0 per timestep, and penalties for control and impact costs.

The episode ends if the humanoid's torso height falls below 1.0 or rises above 2.0, signaling failure or unrealistic behavior. The state observation consists of joint positions, velocities, inertial data, and external forces, providing the necessary information for the agent to control the humanoid in various directions. This modified environment introduces diverse task variations by randomizing the movement direction for each episode, making it suitable for lifelong learning experiments.

F.1.10 HalfCheetah Environment (Gravity and Body-Parts)

The HalfCheetah environment (as shown in Figure 5f) simulates a 2D cheetah robot with multiple joints and a torso, where the agent controls the forces applied to the joints in order to move forward as efficiently as possible. In the gravity domain, each task corresponds to a random gravity value between 0.5g and 1.5g. In the body-parts domain, the size and mass of four body parts (head, torso, thigh, and leg) are randomly scaled between $0.5 \times$ and $1.5 \times$ their nominal values. These task variations allow for highly diverse challenges as the agent must adapt to changes in both external forces and internal dynamics.

F.1.11 Hopper Environment (Gravity and Body-Parts)

The Hopper environment (as shown in Figure 5g) simulates a single-legged robot with joints at the knee and hip, and the goal is to hop forward as efficiently as possible. In the gravity domain, each task is defined by a random gravity value between 0.5g and 1.5g. In the body-parts domain, the size and mass of four body parts (head, torso, thigh, and leg) are randomly scaled between $0.5 \times$ and $1.5 \times$ their nominal values. These task variations introduce significant changes in the robot's behavior, requiring the agent to adapt its hopping strategy to different environmental and physical dynamics.

F.1.12 Walker-2D Environment (Gravity and Body-Parts)

The Walker-2D environment (as shown in Figure 5h) simulates a bipedal robot walking on a plane. The agent must control the forces applied to the legs and torso to walk forward without falling. In the gravity domain, tasks are defined by random gravity values between 0.5g and 1.5g. In the body-parts domain, the size and mass of the head, torso, thigh, and leg are scaled randomly between $0.5 \times$ and $1.5 \times$ their nominal values. These variations create diverse and challenging tasks where the agent must adjust its gait and posture to compensate for different physical properties.

F.2 Hyper-Parameters

Table 2 list hyperparameters used in EPICG. Among these hyperparameters, the frequency of lifelong update, i.e., N is very important and closely related to the performance of both algorithm. Therefore, N is choosen carefuly

for each environment, whose values are shown in Table 3. For hyperparameters of other methods, we use the original source code with parameters and model architectures suggested in the original paper. The experiments were done in the GeForce RTX 2080 Ti GPU with 10 GB Memories.

Hyperparameters	Values
taks (K)	2000 or 1000
learning rate	10^{-4}
β	10^{-4}
N	chosen the best from $\{5, 10, 25, 50\}$
initial value of λ	0.9
decay factor of λ	0.95

Table 2:	Hyparameters	of EPICG
----------	--------------	----------

EPICG
25
25
25
25
25
10
25
10
10
10
25
25
25
25

Table 3: Lifelong update frequency of EPICG