Lifelong Incremental Reinforcement Learning With Online Bayesian Inference

Zhi Wang, *Member, IEEE*, Chunlin Chen, *Member, IEEE*, and Daoyi Dong, *Senior Member, IEEE*

***Abstract*— A central capability of a long-lived reinforcement learning (RL) agent is to incrementally adapt its behavior as its environment changes and to incrementally build upon previous experiences to facilitate future learning in real-world scenarios. In this article, we propose lifelong incremental reinforcement learning (LLIRL), a new incremental algorithm for efficient lifelong adaptation to dynamic environments. We develop and maintain a library that contains an infinite mixture of parameterized environment models, which is equivalent to clustering environment parameters in a latent space. The prior distribution over the mixture is formulated as a Chinese restaurant process (CRP), which incrementally instantiates new environment models without any external information to signal environmental changes in advance. During lifelong learning, we employ the expectation–maximization (EM) algorithm with online Bayesian inference to update the mixture in a fully incremental manner. In EM, the E-step involves estimating the posterior expectation of environment-to-cluster assignments, whereas the M-step updates the environment parameters for future learning. This method allows for all environment models to be adapted as necessary, with new models instantiated for environmental changes and old models retrieved when previously seen environments are encountered again. Simulation experiments demonstrate that LLIRL outperforms relevant existing methods and enables effective incremental adaptation to various dynamic environments for lifelong learning.**

***Index Terms*— Bayesian inference, Chinese restaurant process (CRP), expectation–maximization (EM), incremental reinforcement learning (RL), lifelong learning.**

I. Introduction

REINFORCEMENT learning (RL) [1] is a kind of algorithms that permits an autonomous active agent to adapt its behavior in a trial-and-error manner to maximize cumulative reward during interaction with an initially unknown environment. Classical algorithms, such as dynamic programming [2], Monte Carlo methods [3], and temporal-difference learning [4], have been successfully applied to Markov decision processes (MDPs) with discrete state-action spaces, even when the reward feedback is sparse or delayed [5], [6]. To overcome the "curse of dimensionality," function approximation techniques liberate RL from traditional tabular algorithms that usually converge slowly with unaffordable computational costs, making RL applicable for MDPs with large or continuous state-action spaces [7], [8]. The recent partnership with deep learning, referred to as deep reinforcement learning (DRL), makes RL being capable of solving extremely high-dimensional problems ranging from video games [9], [10] and board games [11] to robotic control tasks [12], [13].

RL methods generally operate in a "stationary" regime: all training is performed in advance, producing policies to make decisions at test time in settings that approximately match those seen during training. However, the environment is often dynamic in real-world scenarios where the reward or state transition functions or even the state-action spaces may change over time. Sudden changes and dynamic uncertainties [14], such as shifts in the terrain for robot navigation [15] or variation in coexisting agents for multiagent systems [16], can cause conventional learning algorithms to fail. Since intelligent agents are becoming ubiquitous with human interactions, an increasing number of scenarios require new learning mechanisms that are amenable for fast adaptation to environments that may drift or change from their nominal situations [17]. A central ability of a long-lived autonomous RL agent is to incrementally adapt its behavior as the environment changes around it, continuously exploiting previous knowledge to facilitate its lifelong learning procedure. Unfortunately, these requirements can be problematic for many established RL algorithms.

Recently, incremental RL [18], [19] emerges as an effective alternative for fast adaptation to dynamic environments.[1] In this setting, the dynamic environment can be considered as a sequence of stationary tasks on a certain timescale where each task corresponds to the specific environmental characteristics during the associated time period. As shown in Fig. 1(a), the previously learned knowledge (e.g., value functions or policies) is utilized for initialization of the new learning process whenever the environment changes, and subsequently, it is adjusted to a new one that fits in the new environment in an incremental manner. Such incremental adaptation is crucial

Manuscript received 12 May 2020; revised 24 August 2020 and 7 December 2020; accepted 26 January 2021. Date of publication 11 February 2021; date of current version 4 August 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 62006111, Grant 62073160, and Grant 61828303; in part by the Australian Research Council's Discovery Projects Funding Scheme under Project DP190101566; in part by the Natural Science Foundation of Jiangsu Province of China under Grant BK20200330; and in part by the Fundamental Research Funds for the Central Universities of China under Grant XJ2020003201. *(Corresponding author: Chunlin Chen.)*

Zhi Wang and Chunlin Chen are with the Department of Control and Systems Engineering, School of Management and Engineering, Nanjing University, Nanjing 210093, China (e-mail: zhiwang@nju.edu.cn; clchen@nju.edu.cn).

Daoyi Dong is with the School of Engineering and Information Technology, University of New South Wales, Canberra, ACT 2600, Australia (e-mail: daoyidong@gmail.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3055499.

Digital Object Identifier 10.1109/TNNLS.2021.3055499

[1] Moreover, incremental learning has been widely investigated to cope with learning tasks with an incoming stream of data or an ever-changing environment [20], in various areas including supervised learning [21], machine vision [22], evolutionary computation [23], human–robot interaction [24], and system modeling [25].

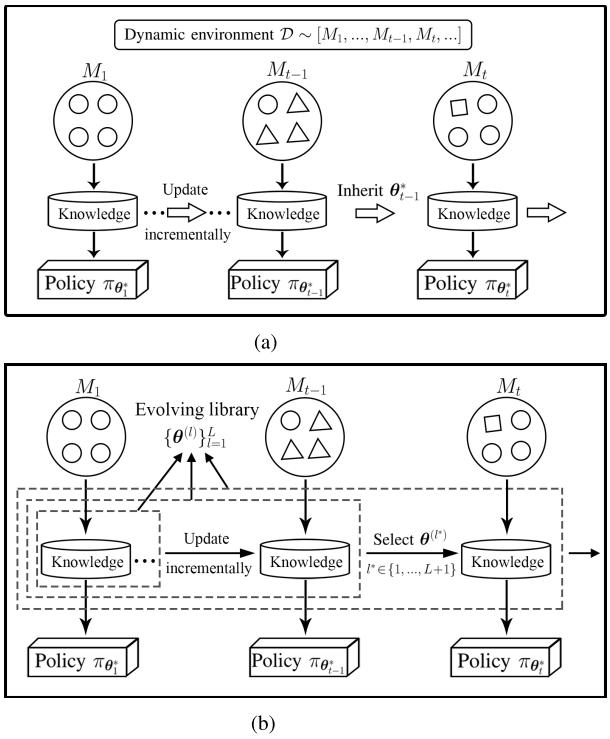


Fig. 1. Comparison between (a) incremental RL and (b) lifelong incremental RL. $M_t \in \mathcal{M}, t = 1, 2, \ldots$ denotes the specific MDP/environment at time period t, $\mathcal{D}$ denotes the dynamic environment over the MDP space $\mathcal{M}$, and $\boldsymbol{\theta}$ denote learning parameters.

for intelligent systems operating in the real world, where changing factors and unexpected perturbations are the norms. For the sake of computational efficiency, Wang *et al.* [18], [19] directly inherited the knowledge from the last time period and discarded all experiences prior to that, thus avoiding repeatedly accessing or processing a large set of previously seen environments. On the other side, it is supposed to be more rational to remember all these experiences as an evolving library during the "lifelong" learning process, as shown in Fig. 1(b). To achieve artificial general intelligence, RL agents should constantly build more complex skills and scaffold their knowledge about the world without forgetting what has already been learned [26]. At a new time period, the learning agent can consult the stored library first and either retrieve the most similar experience (previously seen environment) from the library or expand a new experience (encountering a new environment) into the library.

The goal of this article is to develop a new incremental RL algorithm for lifelong adaptation to dynamic environments. We focus on the way how we selectively retrieve prior experience from the lifelong learning library to help the current learning process most. This work is orthogonal and complementary to the previous one in [18] and [19] where the emphasis is put on how the learned knowledge is fast adapted to a new environment after simply inheriting it from the last time period.

We develop and maintain a library that contains a potentially infinite number of pairwise parameters. One is the canonical "learning parameters" for learning the behavior policy, such as the policy network in direct policy search [12]. The other, denoted as "environment parameters," is to parameterize the environment using an arbitrary function approximator such as a neural network, which can be instantiated as the reward or state transition function. To handle dynamic environment distributions over time, we introduce an infinite mixture of Bayesian models over environment parameters, which is equivalent to clustering environment parameters in a latent space. The prior distribution over the mixture is formulated as a Chinese restaurant process (CRP), where new environment models are sequentially instantiated as needed. By using latent variables in a probabilistic mixture model to indicate the environment-to-cluster assignments, we can directly detect similarities between environment models based on the environment-specific likelihood, without requiring environment delineations to be specified in advance. During lifelong learning, we employ the expectation–maximization (EM) algorithm with online Bayesian inference to update the mixture of environment models in a fully incremental manner. The E-step in EM corresponds to computing the posterior inference of environment probabilities, whereas the M-step is amenable for updating environment parameters incrementally for future learning. This allows for all environment models to be adapted as necessary, with new models instantiated for environmental changes and old models retrieved when previously seen environments are encountered again.

The primary contribution of this article is a lifelong incremental reinforcement learning (LLIRL) algorithm that employs EM, in conjunction with a CRP prior on the environment distribution, to learn a mixture of environment models to handle dynamic environments over time. The infinite mixture enables incremental assignments of soft environment-to-cluster probabilities, allowing for environment specialization to emerge naturally without any external information to signal environmental changes in advance. Experiments are conducted on a suite of continuous control tasks ranging from robot navigation to locomotion in various dynamic environments. Our results verify that LLIRL instantiates new environment models as necessary, correctly clusters previously seen environments in a latent space, and incrementally builds upon previous experiences to facilitate adaptation to challenging dynamic environments during lifelong learning.

The remainder of this article is organized as follows. Section II introduces the preliminaries, including RL algorithms and related work. In Section III, we first present the problem statement and the overview of LLIRL, followed by specific implementations in detail and the final integrated algorithm. Experiments on several robot navigation and Mujoco locomotion tasks are conducted in Section IV. Section V presents concluding remarks.

II. Preliminaries and Related Work

A. Reinforcement Learning

RL is commonly studied based on the MDP framework. An MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition probability, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and γ is the discount factor. A policy is defined as

a function $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, a probability distribution that maps actions to states, and $\sum_{a\in\mathcal{A}} \pi(a|s) = 1, \forall s \in \mathcal{S}$. The goal of RL is to find an optimal policy π^* that maximizes the expected long-term return $J(\pi)$

$$J(\pi) = \mathbb{E}_{\tau\sim\pi(\tau)}[r(\tau)] = \mathbb{E}_{\tau\sim\pi(\tau)}\left[\sum_{i=0}^{\infty} \gamma^i r_i\right] \tag{1}$$

where $\tau = (s_0, a_0, s_1, a_1, \ldots)$ is the learning episode, $\pi(\tau) = p(s_0)\Pi_{i=0}^{\infty}\pi(a_i|s_i)p(s_{i+1}|s_i, a_i)$, and r_i is the instant reward received when executing the action a_i in the state s_i.

The policy can be represented as a parameterized approximation $\pi_{\boldsymbol{\theta}}$ using a function $h(\cdot|\boldsymbol{\theta})$. In DRL [12], h is a deep neural network (DNN) and $\boldsymbol{\theta}$ denote its weights. The Gibbs distribution is commonly used for a discrete action space

$$\pi_{\boldsymbol{\theta}}(i|s) = \frac{\exp(h_i(s|\boldsymbol{\theta}))}{\sum_{j\in A(s)} \exp(h_j(s|\boldsymbol{\theta}))} \tag{2}$$

and the Gaussian distribution is usually used for a continuous action space

$$\pi_{\boldsymbol{\theta}}(a|s) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{\sigma^2}(h(s|\boldsymbol{\theta}) - a)^2\right). \tag{3}$$

To measure the quality of the policy π, the direct objective function can be equivalently rewritten as

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\tau\sim\pi_{\boldsymbol{\theta}}(\tau)}[r(\tau)] = \int_{\tau} \pi_{\boldsymbol{\theta}}(\tau) r(\tau)\, d\tau \tag{4}$$

where $r(\tau) = \sum_{i=0}^{\infty} \gamma^i r_i$ is the return of episode τ.

The objective function is commonly maximized by ascending the parameters following the gradient of the policy with respect to the expected return. By the policy gradient theorem [1], the basic policy gradient method employs the direct gradient of the objective

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \mathbb{E}_{\tau\sim\pi_{\boldsymbol{\theta}}(\tau)}\left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\tau) r(\tau)\right] \\ &= \int_{\tau} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\tau) r(\tau) \pi_{\boldsymbol{\theta}}(\tau)\, d\tau \\ &\approx \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\tau^i) r(\tau^i)\end{aligned} \tag{5}$$

where $(\tau^1, \ldots, \tau^m)$ is a batch of learning episodes sampled from policy $\pi_{\boldsymbol{\theta}}$. Hereafter, an ascent step is taken in the direction of the estimated gradient as $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. This process continues until $\boldsymbol{\theta}$ converge [12], [27].

B. Related Work

Incremental learning is related to online learning [28] and continual learning [29], which also consider a sequential setting where tasks are revealed one after another. One of the most representative algorithms for online learning is to follow the leader (FTL) [30], which consolidates all the data from previous tasks into a single large data set and fits a single model to it. Online learning offers an appealing theoretical framework [31] that aims at zero-shot generalization without any task-specific adaptation, while our lifelong incremental learning considers how past experiences can facilitate the learning adaptation to a new task. On the other side, continual learning systems aim to learn a sequence of tasks one by one such that the learning of each new task will not forget how to perform previously trained tasks [32], i.e., mitigating catastrophic forgetting [33]. In contrast, our lifelong incremental learning exploits past experiences in a sequential manner to learn good priors, while it has the ability to rapidly adapt to the current learning task at hand.

Our work is also related to Bayesian policy reuse (BPR) [17] that employs the Bayesian inference to select prior knowledge from a preestablished library. Deep BPR+ [16] extended BPR with DRL techniques to handle nonstationary opponents in multiagent RL. Yang *et al.* [34] incorporated the theory of mind into BPR to detect nonstationary and more sophisticated opponents and to compute the best response accordingly. BPR methods prefer to quickly select a near-optimal policy from a collection of prelearned behaviors that have been acquired offline, while we emphasize optimal adaptation to the ever-changing environment and synchronously incorporates a mixture of Bayesian models to update the library incrementally. Moreover, BPR methods measure task similarities based on the received reward signal, whereas our method is based on the approximated reward or state transition function that exhibits better representation capabilities than the reward signal itself.

Developing smart agents that are able to work under dynamic conditions has attracted increasing attention in the RL community. A particularly related class of methods in the context of dynamic environments is transfer RL [35], which reuses the knowledge from a set of related source domains to help the target learning task. One feasible approach is to use *domain randomization* to train a *robust* policy that can work under a large variety of environments [36]–[39]. This approach relies on task-specific knowledge to schedule the range of randomized domains, while it is usually challenging to balance the range of domains. In contrast, our method provides a flexible structure in which the scale of the mixture model is determined by the observed dynamic environment itself, without any requirement on the range of task distributions.

Instead of learning invariance to environment dynamics, an alternative solution is to train an *adaptive* policy that is able to identify environmental dynamics and apply actions appropriate for different dynamics [40]. Chen *et al.* [41] used a representation of hardware variations as an additional input to the policy function for each discrete instance of the environment. Peng *et al.* [42] and Andrychowicz *et al.* [43] learned adaptive behavior and implicit system identification simultaneously by embedding the summary of past states and actions into a memory-augmented recurrent policy. Adaptive policies can be learned exclusively from the assumed source tasks and applied directly to unknown environments without any additional training. However, policies trained over a source distribution may not generalize well when the discrepancy between the target environment and the source is too large. In contrast, our method incrementally updates and expands a mixture model to handle dynamic environments on the fly, regardless of such discrepancy.

Another line of research that tackles the learning problem in dynamic environments is meta learning [44]. A recent

trend in meta learning is to learn a base model from which adaptation can be quickly performed to new tasks sampled from a fixed distribution. One such approach is the model-agnostic meta learning (MAML) [45], [46], a simple yet elegant meta-learning framework that has achieved state-of-the-art results in a number of settings [47], [48]. In general, existing approaches need to repeatedly access and process a potentially large distribution of training tasks to yield a reliable knowledge base for target environments that are supposed to be consistent with the training distribution. In contrast, our method concentrates on the ability to rapidly learn and adapt in a sequential manner by maintaining a library from scratch, without any structural assumptions or prior knowledge on the dynamics of the ever-changing environment.

III. LifeLong Incremental Reinforcement Learning

In this section, we first formulate the lifelong incremental learning problem in the context of dynamic environments. Then, we introduce the overview of LLIRL that enables the agent to incrementally accumulate knowledge over a lifetime of experience and rapidly adapt to dynamic environments by building upon prior knowledge. Next, we explain in detail the infinite mixture model that formulates the prior distribution on an incrementally increasing number of environment clusters and the EM algorithm with online Bayesian inference to update the mixture of environment models in a fully incremental manner. Finally, we present the integrated LLIRL algorithm based on the above implementations.

A. Problem Formulation

We consider the dynamic environment as a sequence of stationary tasks on a certain timescale where each task corresponds to the specific environment characteristics at the associated time period. Assume that there is a space of MDPs, $\mathcal{M}$, and an infinite sequence of environments, $\mathcal{D}$, over time in $\mathcal{M}$. An RL agent interacts with the dynamic environment $\mathcal{D} = [M_1, \ldots, M_{t-1}, M_t, \ldots]$, where each $M_t \in \mathcal{M}$ denotes the specific MDP/environment that is stationary at the tth time period. The environment changes over time, resulting in a nonstationary environment distribution, and the identity of the current environment M_t is unknown to the agent. We assume in this article that the environment changes only in the reward and state transition functions but keeps the same state and action spaces. The goal of lifelong incremental learning is to build upon the prior knowledge accumulated along with previous time periods $1, 2, \ldots, t-1$, to facilitate optimizing the learning parameters that can achieve maximum return at the current environment M_t as

$$\boldsymbol{\theta}_t^* = \arg\max_{\boldsymbol{\theta} \in \mathbb{R}^d} J_{M_t}(\boldsymbol{\theta}). \tag{6}$$

In an incremental manner, the agent learns optimal parameters $(\boldsymbol{\theta}_{t+1}^*, \boldsymbol{\theta}_{t+2}^*, \ldots)$ over its lifetime, in conjunction with updating the prior knowledge for future learning.

B. Method Overview

A straightforward approach for leveraging prior knowledge is to store every learning instantiation encountered in the past, while it suffers from scalability problems as the number of instantiated environments quickly becomes large. Hence, we start with a more rational idea that parameterizes environment instantiations and clusters previously seen environments in a latent space, reducing redundancy within the stored library.

We develop and maintain a library that contains a potentially infinite number of pairwise parameters $(\boldsymbol{\theta}^{(\infty)}, \boldsymbol{\vartheta}^{(\infty)})$ during the lifelong learning process in a dynamic environment: $\boldsymbol{\theta}$ for learning the behavior policy (e.g., policy network) and $\boldsymbol{\vartheta}$ for parameterizing the environment (e.g., reward or state transition function approximated by a neural network). At time period t, suppose that the accumulated knowledge along with previous time periods $1, 2, \ldots, t-1$ is represented by the library containing L sets of pairwise parameters $\{\boldsymbol{\theta}_t^{(l)}, \boldsymbol{\vartheta}_t^{(l)}\}_{l=1}^{L}$, where $\boldsymbol{\theta}^{(l)}$ and $\boldsymbol{\vartheta}^{(l)}$ denote the learning and environment parameters corresponding to a specific environment cluster $M^{(l)} \in \mathcal{M}$, respectively. The agent should first estimate the identity of the current environment M_t (which is unknown) as

$$z_t = l^*, \quad l^* \in \{1, \ldots, L, L+1\} \tag{7}$$

where z_t is a categorical latent variable indicating the cluster assignment of the environment-specific parameters $\boldsymbol{\vartheta}_t$. $l^* \leq L$ indicates retrieving the most similar model of previously seen environment in the library, and $l^* = L + 1$ indicates the incremental expansion of a new environment cluster into the library. After the environment identification, the agent will initialize learning parameters of the current environment from the library as $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_t^{(l^*)}$, which is considered to help the current learning process most. The learning parameters are further optimized through interacting with the current environment and in turn are used to update the library for future learning as $\boldsymbol{\theta}_{t+1}^{(l^*)} \leftarrow \boldsymbol{\theta}_t^*$.

To handle dynamic environments, we introduce an infinite mixture over the environment-specific parameters $\boldsymbol{\vartheta}$, which is equivalent to clustering the environment parameters in a latent space. The prior distribution $P(\boldsymbol{\vartheta})$ is formulated via the CRP, which will be discussed in Section III-D. Since the number of environment clusters is unknown, we begin with one environment cluster at the first time period, where $L = 1$, and randomly initialize the pairwise parameters $(\boldsymbol{\theta}_1^{(1)}, \boldsymbol{\vartheta}_1^{(1)})$ in the library. From here, we continuously update the environment-specific parameters to model the true dynamic environment and incrementally instantiate new environment clusters as needed via the CRP. At each time period, to identify the unknown current environment M_t, we will use the introduced mixture to infer the prior and posterior distributions over environment clusters, using these distributions to make predictions and in turn using them to update the environment parameters. Thus, the lifelong incremental learning method can adapt the environment parameters at each time period according to the inferred distributions over an increasing number of environment clusters.

Let $\boldsymbol{x}$ and $\boldsymbol{y}$ denote the input and output with respect to the environment model, respectively, and $(\boldsymbol{X}_t, \boldsymbol{Y}_t)$ be the constructed input–output data set at time period t. During lifelong learning, we employ the EM algorithm to update the Bayesian mixture of environment models in a fully incremental manner

without storing previous samples, which will be described in detail in Section III-E. The E-step in EM involves inferring the latent environment-to-cluster probabilities as

$$P(\boldsymbol{\vartheta}_t|\boldsymbol{Y}_t, \boldsymbol{X}_t) \propto p_{\boldsymbol{\vartheta}_t}(\boldsymbol{Y}_t|\boldsymbol{X}_t)P(\boldsymbol{\vartheta}_t) \tag{8}$$

and the M-step optimizes the expected log-likelihood as

$$\mathcal{L}(\boldsymbol{\vartheta}_t) = \mathbb{E}_{M_t\sim P(\boldsymbol{\vartheta}_t|\boldsymbol{X}_t,\boldsymbol{Y}_t)}[\log p_{\boldsymbol{\vartheta}_t}(\boldsymbol{Y}_t|\boldsymbol{X}_t)]. \tag{9}$$

C. Environment Parameterization

Clustering environments as a mixture in a latent space requires a model that can represent the underlying environment and needs to train the parameterized model in a supervised way. Naturally, we can use the reward function

$$r = g^1_{\boldsymbol{\vartheta}}(s, a) \tag{10}$$

or the state transition function

$$s' = g^2_{\boldsymbol{\vartheta}}(s, a) \tag{11}$$

or the concatenation of the two functions

$$[r, s'] = g^3_{\boldsymbol{\vartheta}}(s, a) \tag{12}$$

to parameterize the environment. In this way, $\boldsymbol{x}$ can be the concatenation of the state and action, and $\boldsymbol{y}$ can be the instant reward or next state or their concatenation. The input–output sample $(\boldsymbol{x}, \boldsymbol{y})$ used to train and update environment models can be constructed from the episodic transition (s, a, r, s') in a canonical RL process.

To obtain a slightly large batch of data for each incremental update, we set the input to be the concatenation of h previous states and actions, given by $\boldsymbol{x}_i = [s_{i-h+1}, a_{i-h+1}, \ldots, s_i, a_i]$, and the output to be the corresponding rewards $\boldsymbol{y}_i = [r_{i-h+1}, \ldots, r_i]$ or next states $\boldsymbol{y}_i = [s_{i-h+2}, \ldots, s_{i+1}]$ or their concatenation $\boldsymbol{y}_i = [r_{i-h+1}, s_{i-h+2}, \ldots, r_i, s_{i+1}]$. Since individual transitions at high frequency can be very noisy, using the consecutive h transitions helps damp out the updates. At time period t, let $p_{\boldsymbol{\vartheta}_t}(\boldsymbol{Y}_t|\boldsymbol{X}_t)$ represent the predictive likelihood of the environment model $\boldsymbol{\vartheta}_t$ on episodic samples $(\boldsymbol{X}_t, \boldsymbol{Y}_t) = \sum_{i=h}^{H}(\boldsymbol{x}_i^t, \boldsymbol{y}_i^t)$, where H is the time horizon of the learning episode. The predictive model represents each sample as an independent Gaussian $\mathcal{N}(\boldsymbol{y}_i^t; g_{\boldsymbol{\vartheta}_t}(\boldsymbol{x}_i^t), \sigma^2)$ such that

$$p_{\boldsymbol{\vartheta}_t}(\boldsymbol{Y}_t|\boldsymbol{X}_t) = \Pi_{i=h}^{H}\mathcal{N}(\boldsymbol{y}_i^t; g_{\boldsymbol{\vartheta}_t}(\boldsymbol{x}_i^t), \sigma^2) \tag{13}$$

where σ^2 is a constant.

D. Infinite Mixture for Dynamic Environments

In the regime of dynamic environments, it is important to add mixture components incrementally to enable specialization of different environment models that constitute the lifelong learning process. We employ an infinite/nonparametric Dirichlet process mixture model (DPMM) [49] to formulate the prior distribution over an increasing number of environment clusters, providing a flexible structure in which the number of environment clusters is determined by the observed dynamic environments.

The instantiation of the DPMM that is well suitable for incremental learning can be described via the CRP [50], a distribution over mixture components that embodies the assumed prior distribution over cluster structures [51], [52]. The CRP can be described by a sequence of customers sitting down at the tables of a Chinese restaurant, where customers sitting at the same table belong to the same cluster. Each customer sits down alone at a new table with probability proportional to a concentration parameter or sits at a previously occupied table with probability proportional to the number of customers already sitting there.

In our case with the CRP formulation, the environment identities are inferred in a sequential manner, while the prior distribution over environment clusters allows a new mixture component to be instantiated with some probability, which is essential for the incremental learning implementation. For a sequence of environments $[M_1, \ldots, M_{t-1}, M_t, \ldots]$, the first environment is assigned to the first cluster. At time period t, the prior distribution, i.e., the expectation of environment-to-cluster assignments, for each cluster $M^{(l)}$ is given by

$$P(\boldsymbol{\vartheta}_t^{(l)}) = P(z_t = l) = \begin{cases} \dfrac{n^{(l)}}{t-1+\zeta}, & l \le L \\ \dfrac{\zeta}{t-1+\zeta}, & l = L+1 \end{cases} \tag{14}$$

where $n^{(l)}$ denotes the number of encountered environments already occupying the cluster $M^{(l)}$ and ζ is a fixed positive concentration hyperparameter that controls the instantiation of new clusters. Considering all previous time periods, the prior probability over all environment clusters becomes

$$P\left(\boldsymbol{\vartheta}_t^{(l)}|\boldsymbol{\vartheta}_{1:t-1}, \zeta\right) = \begin{cases} \dfrac{\sum_{t'=1}^{t-1} P(\boldsymbol{\vartheta}_{t'}^{(l)})}{t-1+\zeta}, & l \le L \\ \dfrac{\zeta}{t-1+\zeta}, & l = L+1 \end{cases} \tag{15}$$

where L indicates the number of nonempty clusters and $l = L + 1$ indicates the potential spawning of a new cluster. This nonparametric formation circumvents the necessity for *a priori* fixed number of clusters, enabling the mixture to unboundedly adapt its complexity along with the evolving complexity of the observed dynamic environment. During lifelong learning, new clusters can be naturally instantiated as needed *in an incremental manner*, without any external information to signal environmental changes in advance.

Remark 1: At one extreme when $\zeta = 0$, there is only one environment cluster all the time. Our method degenerates to the incremental learning setting in [18] that directly inherits the prior knowledge from the last time period and discards all experiences prior to that. When the concentration hyperparameter ζ gets larger, the CRP tends to produce more clusters, which is likely to provide more precise clustering results at the cost of more computational efforts. At the other extreme of $\zeta = \infty$, our method always instantiates a new cluster for each environment, resulting in a one-to-one environment-to-cluster mapping. The learning adaptation performance will degrade poorly since, at each time period, we need to learn from scratch without utilizing any prior knowledge.

E. EM With Online Bayesian Inference

To enable lifelong learning in dynamic environments, we employ the EM algorithm with online Bayesian inference to update the mixture of environment models in a fully incremental manner. In our case, the E-step in EM involves estimating the posterior expectation of environment-to-cluster assignments at the current time period $P(\boldsymbol{\vartheta}_t|\boldsymbol{X}_t, \boldsymbol{Y}_t)$, whereas the M-step involves updating environment parameters $\boldsymbol{\vartheta}_t$ to the new $\boldsymbol{\vartheta}_{t+1}$ incrementally for future learning.

We first estimate the expectations over all $L+1$ clusters (including the potentially new one) considering the environment distribution. The posterior distribution of each environment-to-cluster assignment $P(\boldsymbol{\vartheta}_t^{(l)}|\boldsymbol{X}_t, \boldsymbol{Y}_t)$ can be written as

$$P\left(\boldsymbol{\vartheta}_t^{(l)}|\boldsymbol{X}_t, \boldsymbol{Y}_t\right) \propto p_{\boldsymbol{\vartheta}_t^{(l)}}(\boldsymbol{Y}_t|\boldsymbol{X}_t) P\left(\boldsymbol{\vartheta}_t^{(l)}\right). \tag{16}$$

Combining the prior probability in (15) and the predictive likelihood in (13), the posterior probability distribution over environment clusters can be derived as

$$P\left(\boldsymbol{\vartheta}_t^{(l)}|\boldsymbol{X}_t, \boldsymbol{Y}_t\right) \propto \begin{cases} p_{\boldsymbol{\vartheta}_t^{(l)}}(\boldsymbol{Y}_t|\boldsymbol{X}_t) \displaystyle\sum_{t'=1}^{t-1} P\left(\boldsymbol{\vartheta}_{t'}^{(l)}\right), & l \le L \\ p_{\boldsymbol{\vartheta}_t^{(l)}}(\boldsymbol{Y}_t|\boldsymbol{X}_t)\zeta, & l = L+1. \end{cases} \tag{17}$$

With the estimated posterior $P(\boldsymbol{\vartheta}_t^{(l)}|\boldsymbol{X}_t, \boldsymbol{Y}_t)$, we perform the M-step that optimizes the expected log-likelihood in (9) based on the inferred environment probabilities. Suppose that each environment model starts from the prior parameters $\boldsymbol{\vartheta}_1$, the value of $\boldsymbol{\vartheta}_t$ after taking one gradient update at each time period can be derived by

$$\boldsymbol{\vartheta}_{t+1}^{(l)} = \boldsymbol{\vartheta}_1^{(l)} - \beta \sum_{t'=1}^{t} P\left(\boldsymbol{\vartheta}_{t'}^{(l)}|\boldsymbol{X}_{t'}, \boldsymbol{Y}_{t'}\right) \nabla_{\boldsymbol{\vartheta}_{t'}^{(l)}} \log p_{\boldsymbol{\vartheta}_{t'}^{(l)}}(\boldsymbol{Y}_{t'}|\boldsymbol{X}_{t'}) \tag{18}$$

where β is the learning rate for the EM algorithm. As stated in Section III-A, with our incremental learning setting, the mixture of environment models has already been updated for all previous time periods $1, 2, \ldots, t-1$. We can approximate the update in (18) by incrementally updating previous parameters on samples of the current environment as

$$\boldsymbol{\vartheta}_{t+1}^{(l)} = \boldsymbol{\vartheta}_t^{(l)} - \beta P\left(\boldsymbol{\vartheta}_t^{(l)}|\boldsymbol{X}_t, \boldsymbol{Y}_t\right) \nabla_{\boldsymbol{\vartheta}_t^{(l)}} \log p_{\boldsymbol{\vartheta}_t^{(l)}}(\boldsymbol{Y}_t|\boldsymbol{X}_t) \ \forall l. \tag{19}$$

This procedure circumvents the necessity for storing previously seen samples, yielding a fully streaming, incremental learning algorithm with online Bayesian inference. To fully implement the EM algorithm, we need to repeatedly alternate the E- and M-steps to converge, rolling back the previous gradient update at each iteration [52].

F. Integrated Algorithm

With the above implementations, the complete LLIRL algorithm is summarized as in Algorithm 1. At the first time period $t = 1$, the environment mixture is initialized to contain only one entry $L = 1$, and we randomly initialize the pairwise parameters $(\boldsymbol{\theta}_1^{(1)}, \boldsymbol{\vartheta}_1^{(1)})$ in Line 1. From here, the incremental learning process at each time period t is described as follows.

We first initialize the pairwise parameters $(\boldsymbol{\theta}_t^{(L+1)}, \boldsymbol{\vartheta}_t^{(L+1)})$ that correspond to the new potential environment cluster in Line 3. Since the identity of the current environment M_t is unknown, we employ a uniform behavior policy to collect a few episodic transitions $\mathcal{T}_\mathcal{E} = \sum_i (s_i, a_i, r_i, s_i')$ that can mostly explore the state-action space of the environment in Line 4. From $\mathcal{T}_\mathcal{E}$, we can construct the input–output samples $(\boldsymbol{X}_t, \boldsymbol{Y}_t)$ in Line 5, which will be used to infer the environment identity and to update the environment models. With the samples collected in the current environment, we can compute the predictive likelihood over environment clusters (including the potentially new cluster) $p_{\boldsymbol{\vartheta}_t}(\boldsymbol{Y}_t|\boldsymbol{X}_t)$ in Line 6. Combining this estimated likelihood in (13) and the CRP prior probability in (15), we can infer the posterior probabilities over the mixture of environment models $P(\boldsymbol{\vartheta}_t|\boldsymbol{Y}_t, \boldsymbol{X}_t)$ in Line 7.

The CRP prior assigns a probability of adding a new cluster into the environment mixture, while the Bayesian posterior determines whether to expand the new cluster into the library or not. If the posterior probability of the new potential cluster is greater than those of the L nonempty existing clusters as in Line 8, then this new cluster is incrementally expanded into the library as in Lines 9 and 10. Next, we perform the EM algorithm with online Bayesian inference to update the mixture of environment models incrementally. The E-step recalculates the posterior distribution over environment models in Line 13. The M-step improves the expected log-likelihood in (9) based on the inferred posterior distribution, updating environment parameters $\boldsymbol{\vartheta}$ via gradient descent in Line 14. After alternating the E- and M-steps to converge, we can obtain new environment parameters that are updated incrementally for future learning in Line 16. Based on the updated environment parameters, the identity of the current environment is obtained by computing a maximum *a posteriori* (MAP) estimate on the predictive likelihood as $M_t = M^{(l^*)}$ in Line 17, i.e., selecting the environment model that best fits the current samples $(\boldsymbol{X}_t, \boldsymbol{Y}_t)$. LLIRL does not refine cluster assignments of previously observed environments, circumventing the need for multiple expensive passes over the whole library. Instead, we incrementally infer environment parameters and instantiate new clusters during episodic training based on unbiased estimates of log-likelihood gradients.

After the identification of the current environment, we initialize its learning parameters from those associated with the selected environment cluster as $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_t^{(l^*)}$ in Line 18, which is supposed to help the current learning process most.[2] Finally, the agent continues to optimize the policy through interacting with the current environment in Line 19 and then update the corresponding learning parameters in the library after the current learning process converges in Line 20. Correspondingly, the entire process of LLIRL is illustrated by a flow diagram, as shown in Fig. 2.

[2]When a new cluster is created (i.e., $l^* = L+1$), we can initialize its policy parameters from one of the L preexisting clusters. Empirically, the policy initialized from another environment may achieve better performance than a randomly initialized one because the previous optimum of policy parameters has learned some of feature representations (e.g., nodes in a neural network) of the state-action space [19].

Algorithm 1 LLIRL With Online Bayesian Inference

Input: Dynamic environment $\mathcal{D} = [M_1, \ldots, M_{t-1}, M_t, \ldots]$
Output: Optimal learning parameters $\boldsymbol{\theta}_t^*$ for each time period during lifelong learning

Initialize $L = 1$, $t = 1$, and $(\boldsymbol{\theta}_1^{(1)}, \boldsymbol{\vartheta}_1^{(1)})$
for *each time period* t **do**
  Initialize $(\boldsymbol{\theta}_t^{(L+1)}, \boldsymbol{\vartheta}_t^{(L+1)})$ // for the new potential environment cluster
  Collect a few transitions $\mathcal{T}_\mathcal{E} = \sum_i (s_i, a_i, r_i, s_i')$ // sampled from a uniform behavior policy
  Construct $(\boldsymbol{X}_t, \boldsymbol{Y}_t)$ from $\mathcal{T}_\mathcal{E}$ // samples with respect to the environment models
  Calculate $p_{\boldsymbol{\vartheta}_t^{(l)}}(\boldsymbol{Y}_t|\boldsymbol{X}_t)$ using (13), $\forall l \leq L+1$ // predictive likelihood of environment models on samples
  Infer $P(\boldsymbol{\vartheta}_t^{(l)}|\boldsymbol{X}_t, \boldsymbol{Y}_t)$ using (16), $\forall l \leq L+1$ // posterior of environment-to-cluster assignments
  if $P(\boldsymbol{\vartheta}_t^{(L+1)}|\boldsymbol{X}_t, \boldsymbol{Y}_t) > P(\boldsymbol{\vartheta}_t^{(l)}|\boldsymbol{X}_t, \boldsymbol{Y}_t), \forall l \leq L$ **then**
    Add $(\boldsymbol{\theta}_t^{(L+1)}, \boldsymbol{\vartheta}_t^{(L+1)})$ to $(\boldsymbol{\theta}_t, \boldsymbol{\vartheta}_t)$ thereafter // incremental expansion of the new environment cluster
    $L \leftarrow L + 1$
  end
  while *not converging* **do**
    Re-calculate $P(\boldsymbol{\vartheta}_t^{(l)}|\boldsymbol{X}_t, \boldsymbol{Y}_t)$ using (16) with updated $\boldsymbol{\vartheta}_t^{(l)}$, $\forall l \leq L$ // E-step, update the posterior
    Adapt $\boldsymbol{\vartheta}_t^{(l)}$ using (19) with updated $P(\boldsymbol{\vartheta}_t^{(l)}|\boldsymbol{X}_t, \boldsymbol{Y}_t)$, $\forall l \leq L$ // M-step, update environment parameters
  end
  $\boldsymbol{\vartheta}_{t+1}^{(l)} \leftarrow \boldsymbol{\vartheta}_t^{(l)}$, $\forall l \leq L$ // obtain new environment parameters incrementally for future learning
  $l^* = \arg\max_{l \leq L} p_{\boldsymbol{\vartheta}_{t+1}^{(l)}}(\boldsymbol{Y}_t|\boldsymbol{X}_t)$ // obtain the identity of the current environment
  $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_t^{(l^*)}$ // initialize the learning parameters from the most likely environment cluster
  Update $\boldsymbol{\theta}_t$, obtain $\boldsymbol{\theta}_t^*$ // learn in the current environment until it converges
  $\boldsymbol{\theta}_{t+1}^{(l)} \leftarrow \boldsymbol{\theta}_t^{(l)}, \forall l \leq L$; $\boldsymbol{\theta}_{t+1}^{(l^*)} \leftarrow \boldsymbol{\theta}_t^*$ // obtain new learning parameters incrementally for future learning
end

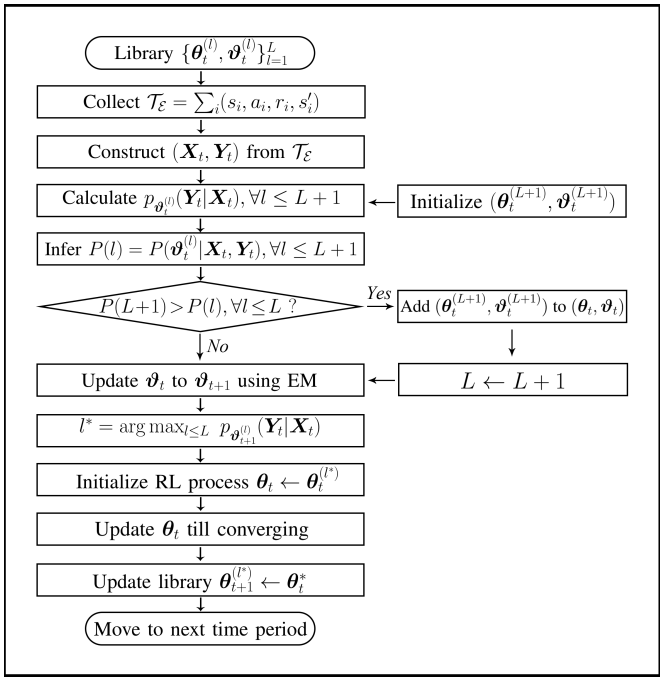


Fig. 2. Flow diagram of LLIRL with online Bayesian inference.

IV. Simulation Experiments

We conduct simulation experiments on continuous control tasks ranging from 2-D navigation to MuJoCo robot locomotion. Using agents in these tasks, we design a number of challenging learning problems that involve infinite (multiple) changes in the underlying environment distribution, where lifelong incremental learning is critical. Through these experiments, we aim to build problem settings that are representative of the types of dynamic environments that RL agents may encounter in real-world scenarios. The overarching questions that we aim to study from our experiments include the following.

- Q1 Can LLIRL handle various dynamic environments where the reward or state transition function may change over the agent's lifetime?
- Q2 Does LLIRL successfully build upon previous experiences to facilitate lifelong learning adaptation to these dynamic environments?
- Q3 How does the number of instantiated environment clusters in the latent space affect the performance?
- Q4 Can LLIRL incrementally instantiate new environment models and correctly cluster these seen environments in a latent space, without any external information to signal environmental changes in advance?

A. Experimental Settings

In Sections IV-B and IV-C, we present the results and insightful analysis of our findings. In the experiments, we evaluate LLIRL in comparison to four baseline methods.

1) *CA:* It continuously adapts a single policy model during lifelong learning. This is representative of commonly used dynamic evaluation methods [46], [53].
2) *Robust:* It takes the most recent observation as the input (i.e., $\pi_{\text{robust}} : s \mapsto a$) and leverages domain randomization to train a robust policy that is supposed to

work for all environments [36], [38], while the current environmental dynamics cannot be identified from its input.

3) *Adaptive:* It represents the policy as a long short-term memory (LSTM) network that takes a history of observations as the input (i.e., $\pi_{\text{adapt}} : [s_{t-l}, \ldots, s_t] \mapsto a$) [42], [43]. This allows the policy to implicitly identify the current environment and adaptively choose actions according to the identified environment.
4) *MAML:* It trains a meta policy by exploiting the dependence between consecutive tasks such that it can solve new learning tasks using only a small number of training samples [45], [47].

We use the policy search algorithm with nonlinear function approximation to handle continuous control tasks [12]. Following the benchmarks [12], we adopt a similar model architecture for all investigated domains. The trained policy of LLIRL is approximated by a feedforward neural network with two 200-unit hidden layers separated by ReLU nonlinearity. The policy network is parameterized by weights $\boldsymbol{\theta}$ and maps each state to the mean of a Gaussian distribution. The environment model is also approximated by a feedforward neural network with two 200-unit hidden layers separated by ReLU nonlinearity, which is parameterized by weights $\boldsymbol{\vartheta}$ and maps each state-action pair to the reward in (10) or the next state in (11) or their concatenation in (12).

For fair comparison to our method, the network architecture of CA, Robust, and MAML is set as the same as that of LLIRL. For Adaptive, we feed a history of five observations to a recurrent policy network that consists of a 200-unit embedding layer and a 200-unit LSTM layer separated by ReLU nonlinearities. The universal policies of Robust, Adaptive, and MAML are trained over a variety of environments that are randomly sampled from a fixed distribution. Furthermore, we continue to train these universal policies after transferring to the new task whenever the environment changes, using the same amount of samples that LLIRL consumes in each environment. We refer to this additional training step as adaptation at execution time. In contrast, LLIRL directly adapts to dynamic environments on the fly without any external information to signal environmental changes in advance, avoiding access to a large distribution of training environments and releasing the dependence on structural assumptions of environmental dynamics.

For each report unit (a particular algorithm running on a particular task), we define two performance metrics. One is the average return over a batch of learning episodes in each policy iteration, which is defined as $(1/m)\sum_{i=1}^{m} r_i(\pi_{\boldsymbol{\theta}})$, where m is the batch size and $r_i(\pi_{\boldsymbol{\theta}})$ is the received return for executing the associated policy. The other is the average return over all policy iterations, which is defined as $(1/mJ)\sum_{j}^{J}\sum_{i=1}^{m} r_i^j(\pi_{\boldsymbol{\theta}})$, where J is the number of training iterations. The former is plotted in figures and the latter is presented in tables. To constitute a lifelong learning process, we sequentially change the environment for $T = 50$ times for each task, resulting in a dynamic environment $\mathcal{D} = [M_1, \ldots, M_T]$. We record the performance of all tested methods for every environment instance $M_t (1 \leq t \leq T)$ and report the statistical results over these T learning adaptation periods to demonstrate the performance of lifelong learning in dynamic environments. All the algorithms are implemented with Python 3.5 running on Ubuntu 16 with 48 Intel Xeon E5-2650 2.20GHz CPU processors, 193-GB RAM, and an NVIDIA Tesla GPU of 32-GB memory. Our code is available online.[3]

B. 2-D Navigation

We first implement LLIRL on a set of navigation tasks where a point agent must move to a goal position within a unit square. The state is the current observation of the 2-D position, and the action corresponds to the 2-D velocity commands that are clipped to be in the range of $[-0.1, 0.1]$. The reward is the negative squared distance to the goal minus a small control cost that is proportional to the action's scale. Each learning episode always starts from a given point and terminates when the agent is within 0.01 of the goal or at the horizon of $H = 100$. The gradient updates are computed using vanilla policy gradient (REINFORCE) [12]. The hyperparameters are set as: learning rates $\alpha = 0.02$ for policy learning and $\beta = 0.001$ for environment model updating, discount factor $\gamma = 0.99$, batch size $m = 16$, and time horizon $h = 4$ for environment parameterization.

1) Representative Types of Dynamic Environments: For Q1, we simulate three representative types of dynamic environments, as shown in Fig. 3.

1) *Type I:* As shown in Fig. 3(a), the dynamic environment is created by changing the goal position within the unit square randomly. Corresponding to the statement in Section III-A, the environment changes in the reward function in this case. To implement the mixture of environment models, we use the reward function as in (10) to parameterize environments.
2) *Type II:* It is a modified version of the benchmark puddle world environment presented in [54] and [55]. As shown in Fig. 3(b), the agent should drive to the goal while avoiding three circular puddles with different sizes. The agent will bounce to its previous position when hitting on the puddles. The dynamic environment is created by moving the puddles within the unit square randomly, i.e., the environment changes in the state transition function, and we use the state transition function as in (11) to parameterize environments.
3) *Type III:* As a combination of the above two types shown in Fig. 3(c), this kind of dynamic environment is created by changing both the goal and puddles within the unit square randomly. The environment changes in both the reward and state transition functions, which is considered to be more complex than the other two types. Corresponding to Section III-C, we use the concatenation of the reward and state transition functions as in (12) to parameterize this type of complex environments.

2) Results of Lifelong Learning Adaptation: To address Q1 and Q2, we present primary results of LLIRL and all baselines implemented on the three types of dynamic environments. Fig. 4 shows the average return per policy iteration, and

[3]https://github.com/HeyuanMingong/llirl

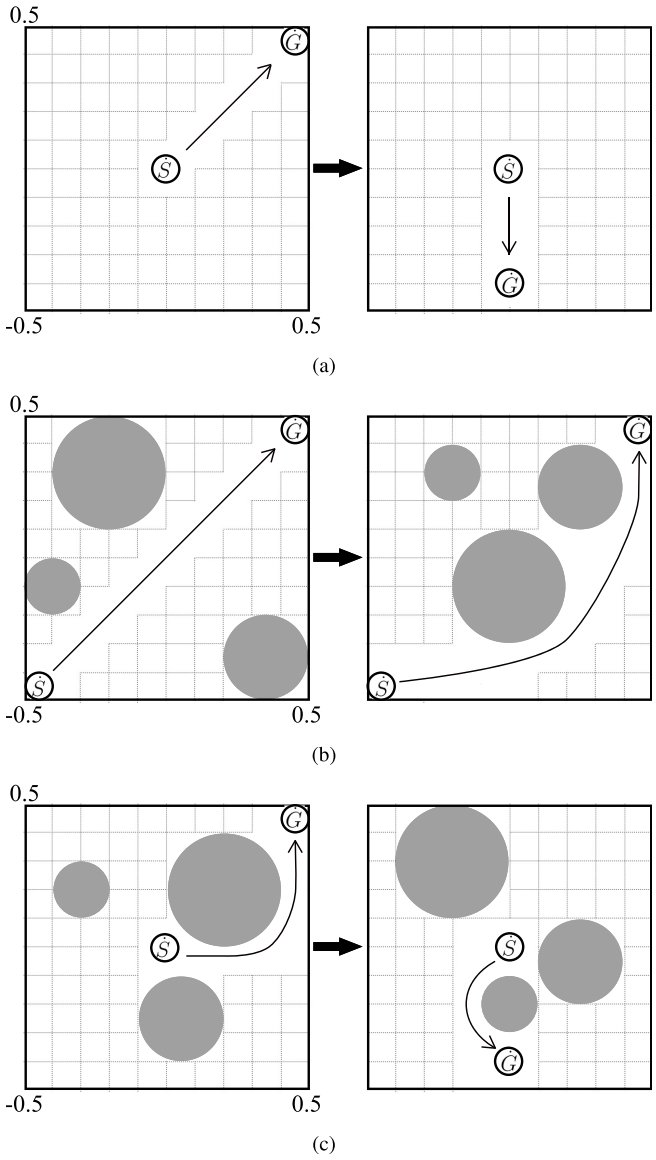


Fig. 3. Examples of three types of dynamic environments in the 2-D navigation tasks. $\dot{S}$ is the start point and $\dot{G}$ is the goal point. Puddles are shown in gray. (a) Type I: the goal changes. (b) Type II: the puddles change. (c) Type III: both the goal and the puddles change.

Table I reports the numerical results in terms of average return over 100 iterations. For LLIRL, the numbers of instantiated clusters are $L = 6$, $L = 4$, and $L = 5$ for the three types of navigation tasks. Obviously, CA obtains the slowest learning adaptation to dynamic environments since it adopts the simplest adaptation mechanism. Robust and Adaptive achieve better performance than CA, which is supposed to benefit from leveraging the domain randomization technique. MAML performs the best among all baselines, exhibiting its ability to embed across-task knowledge into the meta policy and acquire task-specific knowledge quickly at execution time.

From Fig. 4, it can be observed that LLIRL achieves significant jumpstart performance [56] compared to all baselines. Due to correctly clustering encountered environments in a latent space, LLIRL is able to retrieve the most similar experience from the library to help the current learning process most. Furthermore, LLIRL achieves much faster learning adaptation to all dynamic environments compared to the four baselines. For instance, in the type I dynamic environment, it takes only 20 policy iterations for LLIRL to obtain near-optimal asymptotic performance, while it takes more than 100 iterations for all baselines. The performance gap in terms of average return per iteration is more pronounced for smaller amounts of computation, which is supposed to benefit from the distinct acceleration of correctly retrieving the most similar environment cluster from the library. From Table I, it can be obtained that LLIRL receives significantly larger average returns over all training iterations than all baselines. In addition, it can be observed from the statistical results that LLIRL mostly obtains smaller confidence intervals and standard errors than the baselines. It indicates that LLIRL can provide more stable learning adaptation to these dynamic environments. In summary, consistent with the statement in Section III-A, it is verified that LLIRL is capable of handling dynamic environments where the reward or state transition function may change over time, providing significantly better learning adaptation to them.

TABLE I

NUMERICAL RESULTS IN TERMS OF AVERAGE RETURN OVER ALL ITERATIONS OF ALL TESTED METHODS IMPLEMENTED IN THE 2-D NAVIGATION TASKS. HERE AND IN SIMILAR TABLES BELOW, THE MEAN ACROSS $T = 50$ CONSECUTIVE ENVIRONMENTAL CHANGES IS PRESENTED, AND THE CONFIDENCE INTERVALS ARE CORRESPONDING STANDARD ERRORS. THE BEST PERFORMANCE IS MARKED IN BOLDFACE

Task	Type I	Type II	Type III
CA	-33.82 ± 0.79	-33.98 ± 0.88	-32.07 ± 0.80
Robust	-24.26 ± 0.50	-32.64 ± 0.32	-25.69 ± 0.53
Adaptive	-31.56 ± 0.59	-31.41 ± 0.27	-30.49 ± 0.48
MAML	-20.31 ± 0.80	-29.17 ± 0.33	-22.21 ± 0.77
LLIRL	$\mathbf{-11.36 \pm 0.25}$	$\mathbf{-22.46 \pm 0.42}$	$\mathbf{-15.63 \pm 0.25}$

3) Influence of the Number of Clusters: To address Q3, i.e., identifying the relationship between the number of instantiated environment clusters and the performance of LLIRL, we vary hyperparameters of the CRP prior and the EM algorithm to obtain a series of implementations with different numbers of instantiated environment clusters. The performance of various LLIRL implementations with different numbers of clusters in the three types of navigation tasks is shown in Fig. 5 and Table II. At one extreme, LLIRL with only one cluster degenerates to the CA baseline. It can be observed that adding only one cluster (LLIRL with two clusters) is already capable of improving the learning adaptation to a large extent compared to the CA baseline. Imagine an extreme situation where the dynamic environment consists of two opposed tasks that are consecutively switched. Initializing the policy from the last time period probably provides little improvement for the current learning process, while initializing from the second-last time period tends to benefit a lot. In this case, maintaining two clusters of knowledge instead of one will significantly enhance learning adaptation to dynamic environments.

In the beginning, adding several clusters will generally help improve the learning adaptation performance since a library

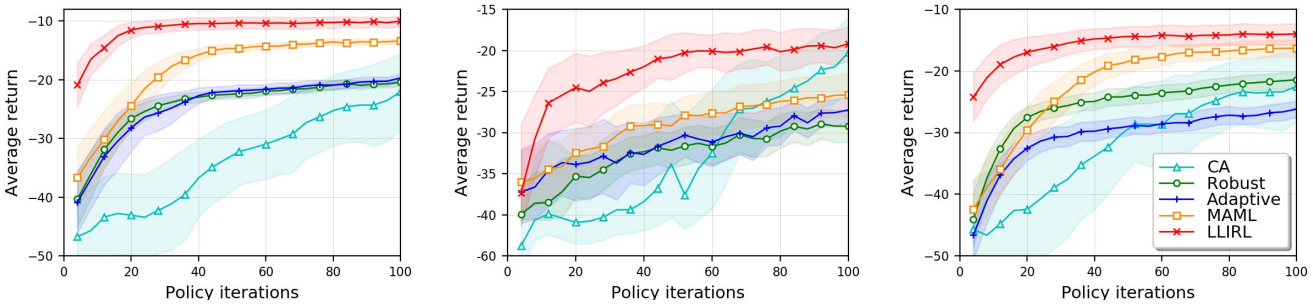


Fig. 4. Average return per iteration of all tested methods in the 2-D navigation tasks. L is the number of instantiated environment clusters by LLIRL. Here and in similar figures below, the mean of average return per iteration across $T = 50$ consecutive environmental changes is plotted as the bold line with 95% bootstrapped confidence intervals of the mean (shaded). (a) Type I, $L = 6$. (b) Type II, $L = 4$. (c) Type III, $L = 5$.

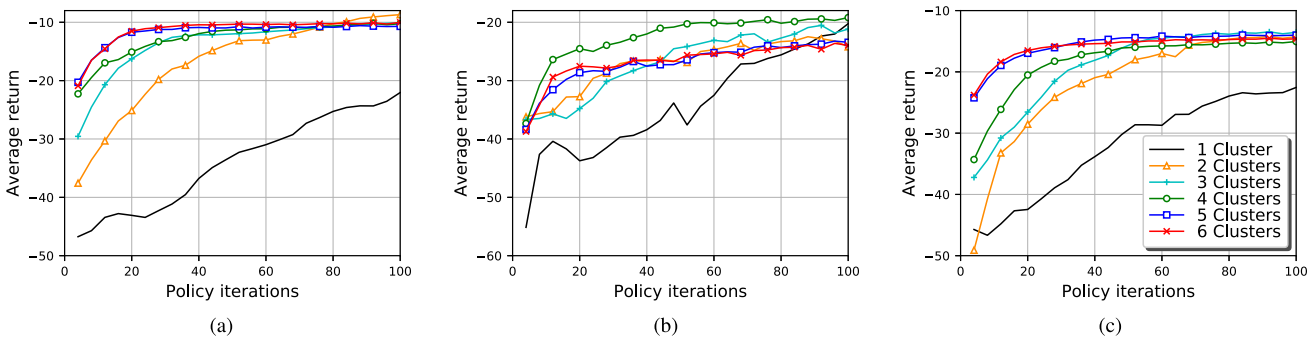


Fig. 5. Average return per iteration of LLIRL implementations with different numbers of instantiated environment clusters in the navigation tasks. (a) Type I. (b) Type II. (c) Type III.

TABLE II

NUMERICAL RESULTS IN TERMS OF AVERAGE RETURN OVER ALL ITERATIONS OF LLIRL IMPLEMENTATIONS WITH DIFFERENT NUMBERS OF INSTANTIATED ENVIRONMENT CLUSTERS IN THE NAVIGATION TASKS

# of clusters	Type I	Type II	Type III
1	-33.82 ± 0.79	-33.98 ± 0.88	-32.07 ± 0.80
2	-16.71 ± 0.81	-27.07 ± 0.43	-21.65 ± 0.90
3	-13.65 ± 0.48	-26.82 ± 0.56	-19.05 ± 0.71
4	-12.70 ± 0.31	$\mathbf{-22.46 \pm 0.42}$	-18.29 ± 0.49
5	-11.73 ± 0.22	-26.95 ± 0.36	$\mathbf{-15.63 \pm 0.25}$
6	$\mathbf{-11.36 \pm 0.25}$	-26.77 ± 0.34	-15.88 ± 0.21

with more clusters of knowledge is likely to provide more appropriate policy initialization for the learning process at each time period. However, as the number of instantiated environment clusters increases, the learning performance is hardly improved and may even be degraded further. At the other extreme, LLIRL will assign each environment to a distinct cluster, resulting in a one-to-one environment-to-cluster mapping. In this case, LLIRL degenerates to the setting that requires learning from scratch whenever the environment changes, thus leading to poor scalability in constantly changing environments. In practice, a moderate number of instantiated environment clusters (e.g., 4–6) are sufficient to obtain appealing performance in these navigation tasks.

4) Incremental Cluster Instantiation and Clustering: To answer Q4, deep insights into the mixture of environment models are required for observing and comprehending the lifelong learning process. We employ the type I navigation task to serve as a proof of principle and a means to gain an intuition of the Bayesian mixture through visualization. The environment is characterized by the reward function that is highly correlated with the goal position. Environments with adjacent goal positions are more similar to each other and tend to belong to the same cluster. Hence, we use the goal position in the 2-D coordinate as a visualization to reveal the relationship among environments.

As shown in Fig. 6, each data point within the unit square stands for a goal position that corresponds to the environment at a specific time period, and environments belonging to different clusters are represented by data points with different shapes and colors. In this implementation, it can be observed that the six environment clusters are incrementally instantiated at time periods $t = 1, 2, 3, 5, 16, 32$, respectively. Eventually, the changing environments over all $T = 50$ time periods are effectively clustered as six components in a latent space, which is visualized in the unit square as shown in Fig. 6(e). More results of LLIRL implementations with different numbers of instantiated environment clusters are presented in Fig. 7. It further verifies that LLIRL is capable of clustering previously seen environments in a latent space where similar environments are close to each other and tend to belong to the

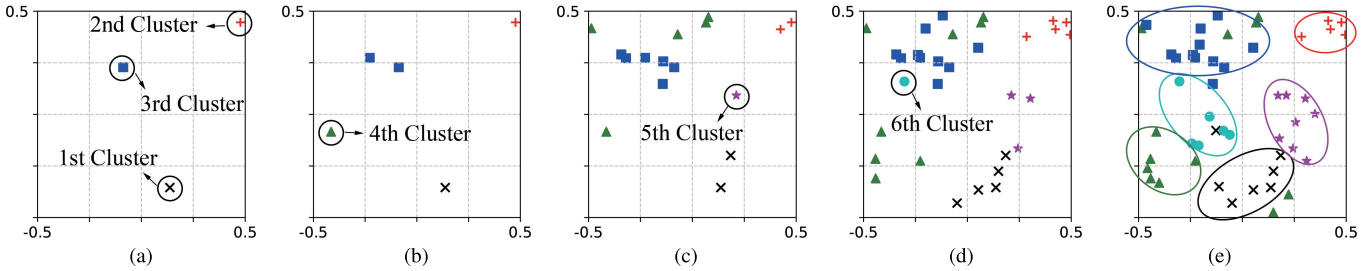


Fig. 6. LLIRL implementation visualized in the 2-D coordinate in type I navigation task. (a) Initial three clusters are instantiated at the initial three time periods. (b)–(d) Fourth-sixth clusters are instantiated at time periods $t = 5, 16, 32$, respectively. (e) All 50 environments are assigned to six clusters effectively.

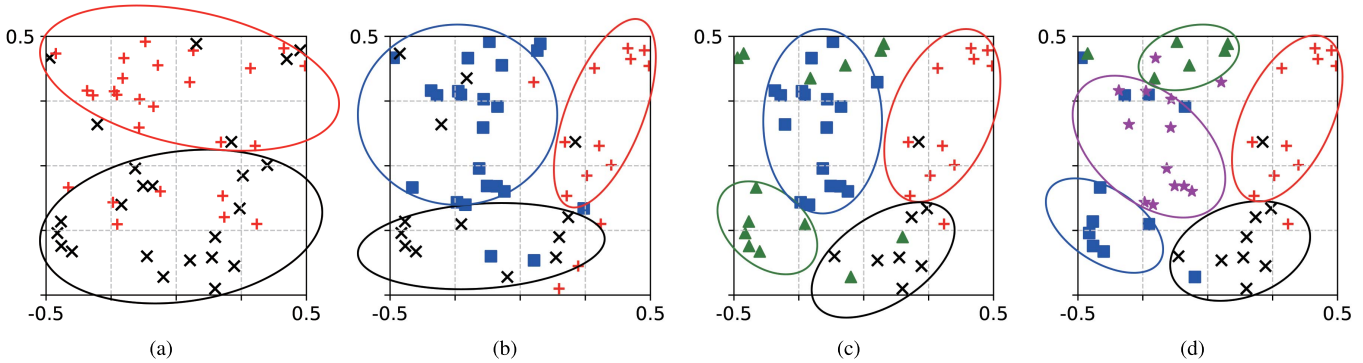


Fig. 7. LLIRL implementations with different numbers of instantiated environment clusters visualized in the 2-D coordinate in type I navigation task. (a) Two clusters. (b) Three clusters. (c) Four clusters. (d) Five clusters.

same cluster. At each time period, the current environment is assigned to an existing cluster or instantiated as a new cluster according to the predictive likelihood of the data samples on these environment clusters. Therefore, no external information is needed to signal environmental changes in advance, which is crucial for lifelong learning in real-world scenarios.

C. MuJoCo Locomotion

The above results verify that LLIRL is well suited to the 2-D navigation tasks, significantly facilitating lifelong learning adaption to various dynamic environments. The next set of experiments is to study whether we can observe similar benefits to lifelong learning when LLIRL is applied to more complex DRL problems. It is necessary to test LLIRL on a well-known problem of considerable difficulty, such as the robotic locomotion control system [57], [58]. Therefore, we also investigate three high-dimensional locomotion tasks with the MuJoCo physics engine [59], aiming at testing whether LLIRL can enable efficient lifelong learning adaptation at the scale of DNNs on much more sophisticated domains.

Fig. 8 shows three representative locomotion tasks with growing dimensions of state-action spaces. These continuous control tasks require a one-legged hopper/planar cheetah/3-D quadruped ant robot to run at a particular velocity in the positive x-direction. The reward is an alive bonus plus a regular part that is negatively correlated with the absolute value between the current velocity of the agent and a goal. The lifelong dynamic environment is created by consecutively

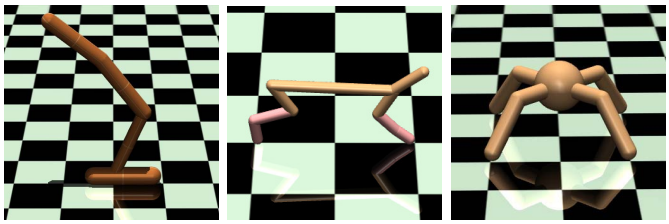


Fig. 8. Representative MuJoCo locomotion tasks with growing dimensions of state-action spaces including (a) Hopper, $|\mathcal{S}| = 11, |\mathcal{A}| = 3$, and $r = 1 - 4 \cdot |v_x - v_g|$, (b) HalfCheetah, $|\mathcal{S}| = 20, |\mathcal{A}| = 6$, and $r = -|v_x - v_g|$, and (c) Ant, $|\mathcal{S}| = 111, |\mathcal{A}| = 8$, and $r = 1 - 3 \cdot |v_x - v_g|$. v_x is the agent's velocity in the positive x-direction and v_g is the goal velocity.

changing the goal velocity at random within a preset range: [0.0, 1.0] for Hopper, [0.0, 2.0] for HalfCheetah, and [0.0, 0.5] for Ant. Each learning episode always starts from a given physical status of the agent and terminates when the agent falls down at the horizon of $H = 100$. We employ proximal policy optimization (PPO) [60] as the base algorithm to handle these challenging locomotion tasks. To reduce the variance of optimization, we subtract the standard linear feature baseline from the empirical return and fit the baseline separately at each policy iteration [12]. Since the environment changes in the reward function, we use the reward function as in (10) to parameterize environments.

With the above settings, we present the results of LLIRL and baseline methods implemented on locomotion domains. For LLIRL, four environment clusters are instantiated for all

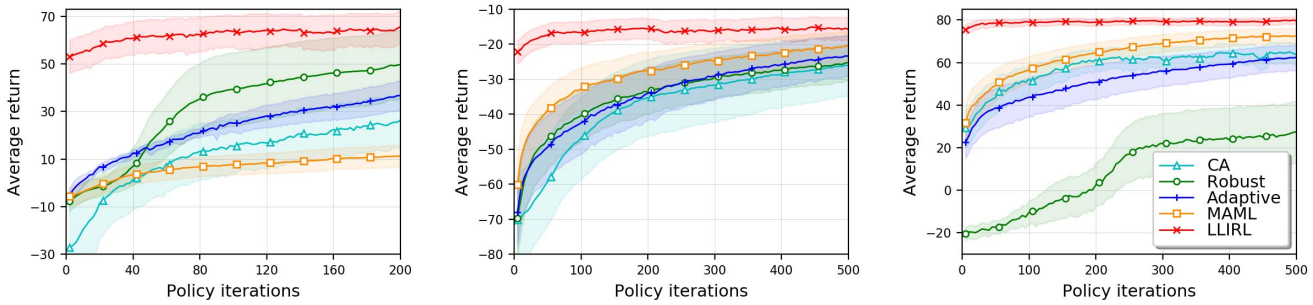


Fig. 9. Average return per iteration of all tested methods in locomotion tasks. $L = 4$ environment clusters are instantiated for LLIRL in all tasks. (a) Hopper. (b) HalfCheetah. (c) Ant.

TABLE III
NUMERICAL RESULTS IN TERMS OF AVERAGE RETURN OVER ALL POLICY ITERATIONS OF ALL TESTED METHODS IMPLEMENTED ON LOCOMOTION TASKS

Task	Hopper	HalfCheetah	Ant
CA	11.19 ± 0.93	-37.70 ± 0.53	57.78 ± 0.38
Robust	30.85 ± 1.30	-34.23 ± 0.40	8.70 ± 0.75
Adaptive	22.39 ± 0.76	-34.13 ± 0.44	51.46 ± 0.40
MAML	6.37 ± 0.30	-28.23 ± 0.34	63.81 ± 1.41
LLIRL	$\mathbf{62.17 \pm 0.19}$	$\mathbf{-16.28 \pm 0.05}$	$\mathbf{79.02 \pm 0.03}$

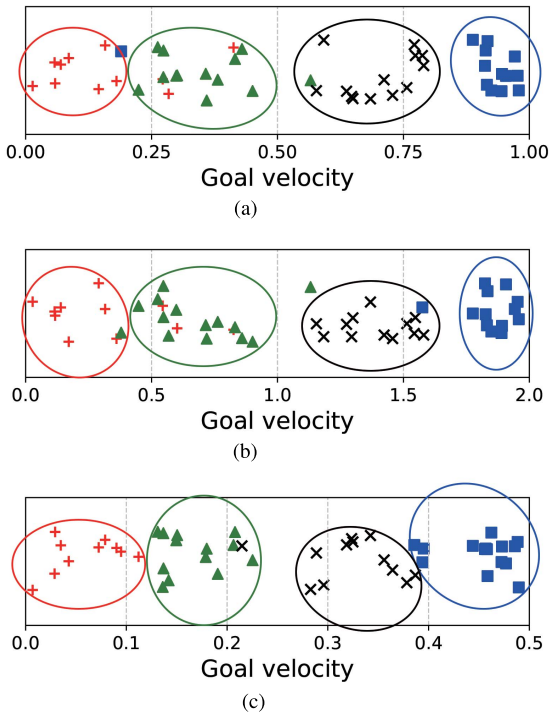


Fig. 10. LLIRL implementations with four instantiated environment clusters in the locomotion tasks. (a) Hopper. (b) HalfCheetah. (c) Ant.

domains. Both LLIRL and baseline methods initialize the policy network in a specific manner and transfer the policy initialization as an inductive bias to help the current learning process. The performance improvement comes from the help of each specific kind of inductive bias, which is empirically evaluated by conducted experiments with some predefined performance metrics. Following the state-of-the-art benchmarks in the RL community [10], [12], [45], [51], we employ the learning curve (i.e., the received return regarding learning iterations) and the average return over all learning iterations as the performance metrics to evaluate all tested methods. Fig. 9 shows the average return per policy iteration of all tested methods, and Table III shows the corresponding numerical results in terms of average return over all policy iterations.

It can be observed that LLIRL always exhibits significantly faster and more stable learning adaptation than all baselines, especially in the initial policy iterations. Actually, LLIRL is already capable of obtaining near-optimal asymptotic performance at the beginning of the learning process whenever the environment changes. In contrast, it takes much more computational efforts for baseline methods to achieve performance comparable with that of LLIRL. For instance, in HalfCheetah and Ant domains, LLIRL only needs approximately 50 learning iterations to receive near-optimal returns, while it can cost more than 500 iterations for baseline methods. More specifically, for obtaining a return of $-20/80$ in the HalfCheetah/Ant domain, LLIRL needs only 13/24 s, while all baselines need more than 260/600 s. This phenomenon reveals that LLIRL successfully builds upon previous experiences to facilitate learning adaptation in dynamic environments to a large extent. CA initializes the policy network directly from the last time period, which has no guaranteed similarity with the current environment. Robust/Adaptive/MAML leverage domain randomization/implicit system identification/meta learning to train a universal policy as the initialization for all environments. These three methods can be considered as transferring the same *averaged* inductive bias to all environments, other than retrieving the most helpful inductive bias for each specific environment. In contrast, LLIRL automatically detects the identity of the current environment under the introduced online Bayesian inference framework. Using the recognized identity, LLIRL retrieves the most similar experience (i.e., learning parameters $\boldsymbol{\theta}$) from the library, which is supposed to maximally benefit the current learning process. Therefore, LLIRL only needs to *finetune* the selected prior knowledge with a small amount of computational resources, being much more efficient for lifelong learning in dynamic environments.

Similar to the analysis in navigation tasks, we also illustrate some deep insights to look into the Bayesian mixture in locomotion domains. Apparently, the locomotion environment is characterized by the reward function that is highly correlated with the goal velocity. Environments with adjacent goal velocities are closer to each other and are more likely to belong to the same cluster. Therefore, we use the goal velocity in the 1-D coordinate to visualize and reveal the relationship among environments. Fig. 10 shows the final clustering results of LLIRL implementations with four instantiated environment clusters in the locomotion tasks. It is once again verified that LLIRL can correctly cluster previously seen environments in a latent space where environments with similar goal velocities tend to belong to the same cluster. This part of clustering using online Bayesian inference is the cornerstone for incrementally building upon previous experiences to enhance lifelong learning adaptation in challenging dynamic environments.

V. Conclusion

In this article, we have presented a lifelong incremental learning framework that adaptively modifies the RL agent's behavior as the environment changes over its lifetime, incrementally building upon previous experiences to facilitate lifelong learning adaptation. LLIRL employs an EM algorithm, in conjunction with a CRP prior, to maintain a mixture of environment models to handle dynamic environments. During lifelong learning, all environment models are adapted as necessary in a fully incremental manner, with new models instantiated for environmental changes and old models retrieved when previously seen environments are encountered again. The CRP prior over an infinite mixture enables new environment models to be incrementally instantiated as needed without any external information to signal environmental changes in advance. Simulations experiments on a suite of continuous control tasks have demonstrated that LLIRL is capable of building upon previous experiences to facilitate lifelong learning adaptation to various dynamic environments. Our results have shown that LLIRL can correctly cluster environments in a latent space, retrieve previously seen environments, and incrementally instantiate new environment clusters as needed.

While we use policy gradient as our evaluation domain, our method is general and can easily be implemented on other RL architectures (e.g., deep Q-networks [9]). A potential direction for future work would be to develop an efficient framework that introduces only one set of parameters to train the policy and parameterize the environment concurrently. Another insightful direction would be to conduct empirical investigation on systematically comparing traditional control methods and recent RL methods in robot locomotion domains [57].

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[2] R. Bellman, "Dynamic programming," *Science*, vol. 153, nos. 37–31, pp. 34–37, 1966.

[3] G. Tesauro and G. R. Galperin, "On-line policy improvement using Monte-Carlo search," in *Proc. Adv. Neural Inf. Process. Syst.*, 1997, pp. 1068–1074.

[4] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[5] B. Luo, D. Liu, T. Huang, and D. Wang, "Model-free optimal tracking control via critic-only Q-learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 10, pp. 2134–2144, Oct. 2016.

[6] J.-A. Li *et al.*, "Quantum reinforcement learning during human decision-making," *Nature Human Behaviour*, vol. 4, no. 3, pp. 294–307, Mar. 2020.

[7] X. Xu, D. Hu, and X. Lu, "Kernel-based least squares policy iteration for reinforcement learning," *IEEE Trans. Neural Netw.*, vol. 18, no. 4, pp. 973–992, Jul. 2007.

[8] H. Li, Q. Zhang, and D. Zhao, "Deep reinforcement learning-based automatic exploration for navigation in unknown environment," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 6, pp. 2064–2076, Jun. 2020.

[9] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[10] O. Vinyals *et al.*, "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, Nov. 2019.

[11] D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.

[12] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1329–1338.

[13] J. Hwangbo *et al.*, "Learning agile and dynamic motor skills for legged robots," *Sci. Robot.*, vol. 4, no. 26, Jan. 2019, Art. no. eaau5872.

[14] H. Liang, G. Liu, H. Zhang, and T. Huang, "Neural-network-based event-triggered adaptive control of nonaffine nonlinear multiagent systems with dynamic uncertainties," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–12, 2020.

[15] M. A. Kareem Jaradat, M. Al-Rousan, and L. Quadan, "Reinforcement based mobile robot navigation in dynamic environment," *Robot. Comput.-Integr. Manuf.*, vol. 27, no. 1, pp. 135–149, Feb. 2011.

[16] Y. Zheng, Z. Meng, J. Hao, Z. Zhang, T. Yang, and C. Fan, "A deep Bayesian policy reuse approach against non-stationary agents," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 962–972.

[17] B. Rosman, M. Hawasly, and S. Ramamoorthy, "Bayesian policy reuse," *Mach. Learn.*, vol. 104, no. 1, pp. 99–127, Jul. 2016.

[18] Z. Wang, C. Chen, H.-X. Li, D. Dong, and T.-J. Tarn, "Incremental reinforcement learning with prioritized sweeping for dynamic environments," *IEEE/ASME Trans. Mechatronics*, vol. 24, no. 2, pp. 621–632, Apr. 2019.

[19] Z. Wang, H.-X. Li, and C. Chen, "Incremental reinforcement learning in continuous spaces via policy relaxation and importance weighting," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 6, pp. 1870–1883, Jun. 2020.

[20] H. He, S. Chen, K. Li, and X. Xu, "Incremental learning from stream data," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 1901–1914, Dec. 2011.

[21] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Trans. Neural Netw.*, vol. 22, no. 10, pp. 1517–1531, Oct. 2011.

[22] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *Int. J. Comput. Vis.*, vol. 77, nos. 1–3, pp. 125–141, May 2008.

[23] S. Yang and X. Yao, "Population-based incremental learning with associative memory for dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 12, no. 5, pp. 542–561, Oct. 2008.

[24] D. Kulić, C. Ott, D. Lee, J. Ishikawa, and Y. Nakamura, "Incremental learning of full body motion primitives and their sequencing through human motion observation," *Int. J. Robot. Res.*, vol. 31, no. 3, pp. 330–345, Mar. 2012.

[25] Z. Wang and H.-X. Li, "Incremental spatiotemporal learning for online modeling of distributed parameter systems," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 49, no. 12, pp. 2612–2622, Dec. 2019.

[26] H. B. Ammar, R. Tutunov, and E. Eaton, "Safe policy search for lifelong reinforcement learning with sublinear regret," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2361–2369.

[27] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, May 1992.

[28] C. Finn, A. Rajeswaran, S. Kakade, and S. Levine, "Online meta-learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1920–1930.

[29] K. James *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.

[30] J. Hannan, "Approximation to Bayes risk in repeated play," in *Contributions to Theory Games*, vol. 3. Princeton, NJ, USA: Princeton Univ. Press, 1957, pp. 97–139.
[31] S. Shalev-Shwartz, "Online learning and online convex optimization," *Found. Trends Mach. Learn.*, vol. 4, no. 2, pp. 107–194, 2011.
[32] G. Zeng, Y. Chen, B. Cui, and S. Yu, "Continual learning of context-dependent processing in neural networks," *Nature Mach. Intell.*, vol. 1, no. 8, pp. 364–372, Aug. 2019.
[33] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2935–2947, Dec. 2018.
[34] T. Yang, J. Hao, Z. Meng, C. Zhang, Y. Zheng, and Z. Zheng, "Towards efficient detection and optimal response against sophisticated opponents," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 623–629.
[35] J. Pan, X. Wang, Y. Cheng, and Q. Yu, "Multisource transfer double DQN based on actor learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2227–2238, Jun. 2018.
[36] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 23–30.
[37] F. Muratore, M. Gienger, and J. Peters, "Assessing transferability from simulation to reality for reinforcement learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Nov. 8, 2019, doi: 10.1109/TPAMI.2019.2952353.
[38] M. Sheckells, G. Garimella, S. Mishra, and M. Kobilarov, "Using data-driven domain randomization to transfer robust control policies to mobile robots," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 3224–3230.
[39] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: From simulation to reality with domain randomization," *IEEE Trans. Robot.*, vol. 36, no. 1, pp. 1–14, Feb. 2020.
[40] W. Yu, C. K. Liu, and G. Turk, "Policy transfer with strategy optimization," in *Proc. Int. Conf. Learn. Represent.*, 2019.
[41] T. Chen, A. Murali, and A. Gupta, "Hardware conditioned policies for multi-robot transfer learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 9333–9344.
[42] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-Real transfer of robotic control with dynamics randomization," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 1–8.
[43] M. Andrychowicz *et al.*, "Learning dexterous in-hand manipulation," *Int. J. Robot. Res.*, vol. 39, no. 1, pp. 3–20, 2020.
[44] M. Andrychowicz *et al.*, "Learning to learn by gradient descent by gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3981–3989.
[45] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.
[46] A. Nagabandi *et al.*, "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2019.
[47] M. Al-Shedivat, T. Bansal, Y. Burda, I. Sutskever, I. Mordatch, and P. Abbeel, "Continuous adaptation via meta-learning in nonstationary and competitive environments," in *Proc. Int. Conf. Learn. Represent.*, 2018.
[48] A. Antoniou, H. Edwards, and A. Storkey, "How to train your MAML," in *Proc. Int. Conf. Learn. Represent.*, 2019.
[49] C. E. Antoniak, "Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems," in *Proc. Ann. Statist.*, 1974, pp. 1152–1174.
[50] J. Pitman, "Combinatorial stochastic processes," Dept. Statistics, UC Berkeley, Berkeley, CA, USA, Tech. Rep. 621, 2002.
[51] Y. Yu, S.-Y. Chen, Q. Da, and Z.-H. Zhou, "Reusable reinforcement learning via shallow trails," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2204–2215, Jun. 2018.
[52] A. Nagabandi, C. Finn, and S. Levine, "Deep online learning via meta-learning: Continual adaptation for model-based RL," in *Proc. Int. Conf. Learn. Represent.*, 2019.
[53] B. Krause, E. Kahembwe, I. Murray, and S. Renals, "Dynamic evaluation of neural sequence models," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2771–2780.
[54] R. S. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," in *Proc. Adv. Neural Inf. Process. Syst.*, 1996, pp. 1038–1044.
[55] A. Tirinzoni, A. Sessa, M. Pirotta, and M. Restelli, "Importance weighted transfer of samples in reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, vol. 80, 2018, pp. 4936–4945.
[56] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, pp. 1633–1685, Jul. 2009.
[57] T. Geijtenbeek, M. van de Panne, and A. F. van der Stappen, "Flexible muscle-based locomotion for bipedal creatures," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 1–11, Nov. 2013.
[58] Y. Pan, P. Du, H. Xue, and H.-K. Lam, "Singularity-free fixed-time fuzzy control for robotic systems with user-defined performance," *IEEE Trans. Fuzzy Syst.*, early access, Jun. 3, 2020, doi: 10.1109/TFUZZ.2020.2999746.
[59] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 5026–5033.
[60] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*. [Online]. Available: http://arxiv.org/abs/1707.06347

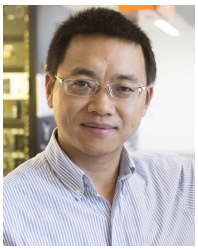
Zhi Wang (Member, IEEE) received the B.E. degree in automation from Nanjing University, Nanjing, China, in 2015, and the Ph.D. degree in machine learning from the Department of Systems Engineering and Engineering Management, City University of Hong Kong, Hong Kong, in 2019.

He had a visiting position at the University of New South Wales, Canberra, ACT, Australia. He is currently an Assistant Professor with the Department of Control and Systems Engineering, Nanjing University. His current research interests include reinforcement learning, machine learning, and robotics.

Chunlin Chen (Member, IEEE) received the B.E. degree in automatic control and the Ph.D. degree in control science and engineering from the University of Science and Technology of China, Hefei, China, in 2001 and 2006, respectively.

He was with the Department of Chemistry, Princeton University, Princeton, NJ, USA, from September 2012 to September 2013. He had visiting positions at the University of New South Wales, Canberra, ACT, Australia, and the City University of Hong Kong, Hong Kong. He is currently a Professor and the Head of the Department of Control and Systems Engineering, School of Management and Engineering, Nanjing University, Nanjing, China. His current research interests include machine learning, intelligent control, and quantum control.

Dr. Chen serves as the Chair for the Technical Committee on Quantum Cybernetics, IEEE Systems, Man and Cybernetics Society.



Daoyi Dong (Senior Member, IEEE) received the B.E. degree and the Ph.D. degree in engineering from the University of Science and Technology of China, Hefei, China, in 2001 and 2006, respectively.

He was with the Chinese Academy of Sciences, Beijing, China, and Zhejiang University, Hangzhou, China. He had visiting positions at Princeton University, Princeton, NJ, USA, RIKEN, Wako, Japan, The University of Hong Kong, Hong Kong, and the University of Duisburg-Essen, Duisburg, Germany. He is currently a Scientia Associate Professor with the University of New South Wales, Canberra, ACT, Australia. He has attracted a number of competitive grants with more than AUS $2.8 million from Australia, USA, China, and Germany. His research interests include machine learning and quantum cybernetics.

Dr. Dong received the ACA Temasek Young Educator Award by The Asian Control Association. He was a recipient of the International Collaboration Award, the Discovery International Award, the Australian Post-Doctoral Fellowship from the Australian Research Council, and the Humboldt Research Fellowship from Alexander von Humboldt Foundation in Germany. He was a co-recipient of the Guan Zhao-Zhi Award at the 34th Chinese Control Conference and the Best Theory Paper Award at the 11th World Congress on Intelligent Control and Automation. He has also served as the general chair or the program chair for several international conferences. He serves as an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS and a Technical Editor for the IEEE/ASME TRANSACTIONS ON MECHATRONICS.