This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Dynamics-Adaptive Continual Reinforcement Learning via Progressive Contextualization

Tiantian Zhang, Zichuan Lin, Yuxing Wang, Deheng Ye, Qiang Fu, Wei Yang, Xueqian Wang, *Member, IEEE*, Bin Liang, *Senior Member, IEEE*, Bo Yuan, *Senior Member, IEEE*, and Xiu Li, *Member, IEEE*

***Abstract*— A key challenge of continual reinforcement learning (CRL) in dynamic environments is to promptly adapt the reinforcement learning (RL) agent's behavior as the environment changes over its lifetime while minimizing the catastrophic forgetting of the learned information. To address this challenge, in this article, we propose DaCoRL, that is, dynamics-adaptive continual RL. DaCoRL learns a context-conditioned policy using progressive contextualization, which incrementally clusters a stream of stationary tasks in the dynamic environment into a series of contexts and opts for an expandable multihead neural network to approximate the policy. Specifically, we define a set of tasks with similar dynamics as an environmental context and formalize context inference as a procedure of online Bayesian infinite Gaussian mixture clustering on environment features, resorting to online Bayesian inference to infer the posterior distribution over contexts. Under the assumption of a Chinese restaurant process (CRP) prior, this technique can accurately classify the current task as a previously seen context or instantiate a new context as needed without relying on any external indicator to signal environmental changes in advance. Furthermore, we employ an expandable multihead neural network whose output layer is synchronously expanded with the newly instantiated context and a knowledge distillation regularization term for retaining the performance on learned tasks. As a general framework that can be coupled with various deep RL algorithms, DaCoRL features consistent superiority over existing methods in terms of stability, overall performance, and generalization ability, as verified by extensive experiments on several robot navigation and MuJoCo locomotion tasks.**

***Index Terms*— Adaptive network expansion, continual reinforcement learning (CRL), dynamic environment, incremental context detection.**

Manuscript received 1 September 2022; revised 24 March 2023; accepted 18 May 2023. This work was supported in part by the Science and Technology Innovation 2030-Key Project under Grant 2021ZD0201404 and in part by the Tencent Rhino-Bird Research Elite Program. *(Corresponding authors: Bo Yuan; Xiu Li.)*

Tiantian Zhang is with the Department of Automation, Tsinghua University, Beijing 100084, China, and also with the Tencent AI Lab, Shenzhen 518000, China (e-mail: ztt19@mails.tsinghua.edu.cn).

Zichuan Lin, Deheng Ye, Qiang Fu, and Wei Yang are with the Tencent AI Lab, Shenzhen 518000, China (e-mail: zichuanlin@tencent.com; dericye@tencent.com; leonfu@tencent.com; willyang@tencent.com).

Yuxing Wang, Xueqian Wang, and Xiu Li are with the Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China (e-mail: wyx20@mails.tsinghua.edu.cn; wang.xq@sz.tsinghua.edu.cn; li.xiu@sz.tsinghua.edu.cn).

Bin Liang is with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: liangbin@mail.tsinghua.edu.cn).

Bo Yuan is with the Research Institute of Tsinghua University in Shenzhen, Shenzhen 518057, China (e-mail: boyuan@ieee.org).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3280085.

Digital Object Identifier 10.1109/TNNLS.2023.3280085

I. Introduction

REINFORCEMENT learning (RL) [1] is a major learning paradigm in machine learning for sequential decision-making tasks. It aims to train a competent policy for an agent that properly maps states to actions to maximize the cumulative reward by interacting with an environment in a trial-and-error manner. Traditional RL algorithms, such as Q-learning [2] and SARSA [3], have been widely studied as tabular methods and successfully applied to Markov decision processes (MDPs) with finite discrete state-action spaces. The emergence of advanced function approximation techniques based on deep neural networks (DNNs) enables RL to have a higher level of understanding of the physical world [4] and solve high-dimensional tasks ranging from playing video games [5], [6], [7], [8], [9], [10] to making real-time decisions on continuous robot control tasks [11], [12].

The progresses of RL have been predominantly focused on learning a single task with the assumption of a stationary[1] and a fully explorable environment for sampling observations. Nevertheless, in the real-world, environments are often nonstationary and characterized by ever-changing dynamics such as shifts in the terrain or weather conditions, changes of the target position in robot navigation [13], and different traffic inflow rates and demand patterns at different times of a day (e.g., peak and off-peak hours) in vehicular traffic signal control [14]. These scenarios demand competent RL agents that can continually adapt to new environmental dynamics while retaining performance when the previous environment is encountered again. Unfortunately, the above requirements are difficult for existing RL methods to fulfill. On the one hand, storing all past experiences may result in a constant growth in memory consumption and computational power. On the other hand, if the size of the replay buffer is limited, the agent may inevitably suffer from the phenomenon known as catastrophic forgetting [15], [16], [17], incapable of retaining the knowledge and skills learned in previously encountered situations.

Continual RL (CRL) [18], [19], [20], [21] has been investigated as an effective solution for adaptation to dynamic environments and mitigation of catastrophic forgetting. In this setting, the dynamic environment can be considered as a stream of stationary tasks on a certain timescale where each task corresponds to the specific environmental dynamics

[1]A stationary environment is an environment whose dynamics normally represented by the reward and state transition functions of the MDP do not change over time.

2162-237X © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Bo Yuan. Downloaded on June 08,2023 at 13:42:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

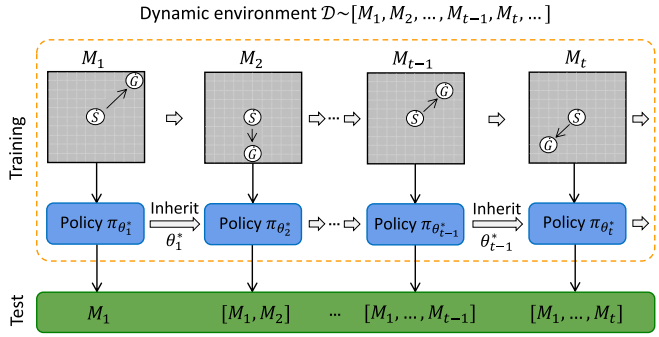


Fig. 1. CRL in dynamic environments. $M_t \in \mathcal{M}, t = 1, 2, \ldots$ denotes the specific MDP/task in time period t; $\mathcal{D}$ denotes the dynamic environment over the MDP space $\mathcal{M}$; θ are the learning parameters of policy; $\pi_{\theta_t^*}$ represents the approximate optimal policy over all learned tasks $[M_1, M_2, \ldots, M_t]$.

during the associated time period. As shown in Fig. 1, the previously learned policy (e.g., $\pi_{\theta_{t-1}^*}$ over $[M_1, M_2, \ldots, M_{t-1}]$) is used for the initialization of the new policy (e.g., π_{θ_t} on M_t), and it is subsequently updated to fit in the current task during the learning period in a continual fashion, retaining previously learned abilities. In other words, CRL is capable of developing proper behaviors for new tasks while keeping the overall performance across all learned tasks. Such features of continual learning are highly desirable for intelligent systems in real-world applications where the environments are subject to consistent changes.

Existing CRL methods assume a decomposition of the original problem into disjoint subdomains of similar dynamics (also called "*context*") and their boundaries are known in advance. Consequently, previous studies mainly focus on how to construct effective mechanisms to mitigate catastrophic forgetting among contexts, largely ignoring the challenge of automatic context inference during the learning process. For the alleviation of catastrophic interference caused by data distribution drift in the single-task RL, Zhang et al. [21] employ sequential K-Means clustering to achieve automatic context inference, given the number of contexts (k) in advance. This method works well because it is feasible to acquire an approximate estimate of the state distribution with sufficient exploration and predetermine the value of k in a single task. However, significant challenges are expected in dynamic environments, where it is impractical to accurately determine the number of environmental contexts in advance as the changes in environmental dynamics are usually infinite and highly uncertain. As a result, it is more rational for the agent to infer and instantiate environmental contexts in a fully online and incremental manner during the CRL process.

In this article, we investigate CRL in dynamic environments to achieve continual context inference and necessary adaptation. The ultimate objective is to maximize the overall performance of the agent in the whole environment. To this end, we propose a novel dynamics-adaptive continual reinforcement learning scheme (DaCoRL) with progressive contextualization, which incrementally clusters a stream of stationary tasks in a dynamic environment into a series of contexts and opts for an expandable multihead neural network to learn a context-conditioned policy. The progressive contextualization contains two core modules: The first one is the incremental context detection procedure for automatically detecting the environmental changes and clustering a set of tasks with similar dynamics into the same context. The second one is the joint optimization procedure to train the policy online for each unique context using an expandable multihead neural network and a knowledge distillation regularization term.

To detect the changes in environmental dynamics over time, we introduce the online Bayesian infinite Gaussian mixture model (IGMM) [22] to cluster environment features in a latent context space, where each cluster corresponds to a separate context. We employ the online Bayesian inference to update the model of contexts in a fully incremental manner, assuming that the prior distribution over the contexts is a CRP [23]. With this online incremental clustering technique, DaCoRL can incrementally instantiate new contexts according to the concentration parameter[2] of CRP as needed, without requiring any external information of environmental changes such as the predetermined number of contexts in [21]. During the joint optimization procedure, we introduce an expandable multihead neural network whose output heads can be adaptively expanded according to the number of instantiated contexts. Compared with fixed-structured networks, this approach can eliminate unnecessary redundancy of network structure and improve learning efficiency.

The contributions of this article are summarized as follows.

1) An incremental context detection strategy is introduced for CRL in dynamic environments. It formalizes context inference as an online Bayesian IGMM clustering procedure, enabling the agent to properly identify changes in environmental dynamics in an online manner without any prior knowledge of contexts.
2) A dynamics-adaptive CRL training scheme called DaCoRL is proposed for dynamic environments in continuous spaces. By employing an expandable multihead neural network and a knowledge distillation regularization term, DaCoRL can effectively alleviate catastrophic forgetting and ensure the competitive capacity of RL agents for continual learning in dynamic environments.
3) Extensive experiments on a suite of continuous control tasks ranging from robot navigation to MuJoCo locomotion are conducted to validate the overall superiority of our method over baselines in terms of stability, overall performance, and generalization ability.

In the rest of this article, Section II reviews the related work and Section III introduces CRL problem statement and relevant concepts. In Section IV, the framework of DaCoRL is presented, with details on its mechanism and implementation. Experimental results and analyses on several robot navigation and MuJoCo locomotion tasks are presented in Section V, and this article is concluded in Section VI with some discussions and directions for future work.

II. Related Work

Continual learning considers how to learn *a sequence of tasks* while maintaining the performance on previously learned tasks. It is conceptually related to incremental learning

[2]It is a hyperparameter in CRP denoted by α in this article to control the likelihood of new contexts.

Authorized licensed use limited to: Bo Yuan. Downloaded on June 08,2023 at 13:42:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

and online learning as they all assume that tasks or samples are presented sequentially. Although incremental learning and continual learning are frequently used interchangeably in the literature, they are not always the same. In some studies [24], [25], [26], incremental learning is used to describe a learning process where a sequence of incremental tasks are learned continually—in this case, continual learning can be referred to as incremental learning. Nevertheless, several other studies on incremental learning [27], [28], [29] concentrate on how to incrementally adjust the previously learned policy to facilitate the fast adaptation to a new task, while not considering the agent performance on old tasks. Online learning aims to fit a single model to *a single task* over a sequence of data instances without any adaptation to new tasks or concerns for the mitigation of catastrophic forgetting [30], [31]. Another related field is multitask learning [32], [33], [34], which tries to train a single model on multiple tasks simultaneously, rather than learning tasks sequentially.

A variety of approaches have been investigated to tackle catastrophic forgetting or interference in the CRL community. They can be classified into three major categories according to how the knowledge of previous tasks is retained and leveraged: replay-based, regularization-based, and expansion-based. The core idea of replay-based approaches is to store samples of past tasks [35], [36], [37] or generate pseudo-samples from a generative model [38] to maintain the knowledge about the past in the model. These previous task samples are replayed while learning a new task in the form of either being reused as model inputs for rehearsal [35], [38] or constraining the optimization of the new task [36], [37], yielding decent results against catastrophic forgetting. An inherent drawback is the constraint on the memory capacity as the number of tasks grows. Although the generative model can be exempted from this limitation, its own training process can also suffer from forgetting. Regularization-based approaches protect learned knowledge from forgetting by imposing an extra regularization term on the learning objective, penalizing large updates on important weights [18], [19] or policies [21], [39], [40] for previous tasks while learning new tasks. This family of solutions is easy to implement and tends to perform well on a small set of tasks, but still faces performance tradeoffs on the new and old tasks as the number of tasks increases. Expansion-based approaches incrementally expand new architectural resources such as the network capacity [41], [42] or a policy library [28], [29], [43] in response to new information. These strategies avoid catastrophic forgetting by protecting all weights for the past tasks from being perturbed by the new information but the knowledge transfer between tasks and the scalability may be limited.

For CRL in dynamic environments, in addition to appropriately adapting the above methods to mitigate catastrophic forgetting, the greatest challenge comes from detecting the changes in the environment autonomously. For instance, regret minimization algorithms can be applied to MDPs with varying transition probability and reward functions, which assume a finite number and degree of MDP changes and detect them based on sliding windows [44], [45], [46], [47] or kernel estimators [48]. In both cases, the algorithms consider a finite horizon and minimize the regret over this horizon, without any concerns for the mitigation of catastrophic forgetting. RLCD [49] is a model-based approach that estimates the prediction quality of different models and instantiates new ones when none of the existing models performs well. An extension of RLCD replaces the quality measure with the CUSUM-based method to perform change-point detection [50]. Unlike our method, these methods are only applicable to purely discrete settings.

Some studies focus on facilitating fast adaptation to new tasks in continuous dynamic environments. In [51] and [29], CUSUM-based change detection mechanisms are combined with well-known RL algorithms, such as REINFORCE [52] and SAC [53], [54], to promote adaptation to new tasks. Nagabandi et al. [55] used meta-learning to train a prior over dynamic models that can, when combined with recent data, be rapidly adapted to the new contexts. This method requires an explicit pretraining phase on all tasks, which is not possible under our CRL settings. LLIRL [28] is a recently proposed method that employs the EM algorithm, together with a CRP prior on the context distribution, to learn an infinite mixture model to cluster the tasks incrementally over time. In this way, it can selectively retrieve previous experiences of the same clusters to facilitate the adaptation to the new task. Similar to [29], LLIRL optimizes two separate sets of network parameters to train the behavior policy and parameterize the environment for each instantiated context, respectively, which greatly increases the training and memory cost. Furthermore, the above studies mainly focus on fast adaptation without considering catastrophic interference among in-context tasks and knowledge transfer among between-context tasks, which are critical issues to be addressed in CRL.

For CRL in dynamic environments, several efforts focus on model-free methods that perform context division directly based on experienced transitions. Context QL [20] applies an ODCP algorithm to state-reward sequences to detect environmental changes. Afterward, it either learns a new Q table for the newly detected context or improves the policy learned if the current context has been previously experienced. This work assumes a preknown pattern and a finite number of changes since ODCP can only determine whether the context has changed, instead of the specific label. Meanwhile, it is only applicable to RL problems with a small and discrete state-action space due to the Q-tables in use. CRL-Unsup [56] detects distributional shifts for continuous dynamic environments by tracking the difference between the short- and long-term moving averages of rewards. When changes are detected, it consolidates the memory using EWC [18] to prevent catastrophic forgetting. This method requires storing all policy weights learned before each EWC process is triggered and can be problematic in the case of a positive forward transfer followed by a negative backward transfer. Recently, IQ [21] shows that performing context division by online clustering and training a multihead neural network with the knowledge distillation technique can alleviate the catastrophic interference caused by data distribution drift in the single-task RL. However, it requires a predetermined number of contexts, which limits its applicability in dynamic environments.

Authorized licensed use limited to: Bo Yuan. Downloaded on June 08,2023 at 13:42:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Our proposed method DaCoRL falls in the combination of regularization-based and expansion-based categories. It performs policy learning using an expandable multihead neural network that has been proved in [57] to be helpful to generalize the knowledge and result in a more effective feature extraction and transfer in multitask learning, avoiding the burden of multiple networks training and storing. To lift the restriction of a predetermined number of contexts, we introduce an incremental context detection module, which can instantiate contexts incrementally as needed. Furthermore, the knowledge distillation technique can effectively reduce the interference among both between-context and in-context tasks, making it possible to conduct effective continual learning in dynamic environments with a single policy network.

III. Preliminaries and Notations

The formulation of the CRL problem in the domain of dynamic environments and the related key concepts are introduced in this section. The notations used in this article are summarized in Appendix A.

A. Problem Formulation

1) RL in Continuous Spaces: RL is commonly studied following the MDP framework [1], which is defined as a tuple $M = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is the set of states; $\mathcal{A}$ is the set of actions; $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition probability function; $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function; and $\gamma \in [0, 1]$ is the discount factor. At each time step $t \in \mathbb{N}$, the agent moves from s_t to s_{t+1} with probability $p(s_{t+1}|s_t, a_t)$ after it takes action a_t and receives instant reward r_t.

Most RL algorithms rely on the mechanism of policy gradient to handle tasks with continuous state and action spaces [58]. In such cases, the policy is defined as a function π: $\mathcal{S} \times \mathcal{A} \to [0, 1]$, mapping each state to a probability distribution of actions, with $\sum_{a \in \mathcal{A}} \pi(a|s) = 1, \forall s \in \mathcal{S}$. If only the state space is continuous, the policy can use the Boltzmann distribution to select discrete actions, while when the action space is also continuous, the Gaussian distribution is commonly used. The goal of policy-based RL is to find an optimal policy π^* with internal parameter $\theta \in \Theta$ that maximizes the expected long-term discount return

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\tau)}\Big[R(\tau)\Big] = \mathbb{E}_{\tau \sim \pi_\theta(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \tag{1}$$

where the expectation is over the complete trajectory τ generated following π_θ until the end of the agent's lifetime.

In basic policy gradient methods (e.g., REINFORCE [52]), the action-selection distribution is usually parameterized by a DNN trained by taking the gradient ascent with the partial derivative of the objective (i.e., maximize the excepted return) with respect to the policy parameters. The policy gradient can be approximately expressed as

$$\begin{aligned} \nabla_\theta J(\theta) &= \nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta(\tau)}\Big[R(\tau)\Big] \\ &= \mathbb{E}_{\tau \sim \pi_\theta(\tau)}\Big[R(\tau) \nabla \log \pi_\theta(\tau)\Big] \\ &\approx \frac{1}{N} \sum_{n=1}^{N} R\big(\tau^n\big) \nabla_\theta \log \pi_\theta\big(\tau^n\big) \end{aligned} \tag{2}$$

where $(\tau^1, \tau^2, \ldots, \tau^N)$ is a batch of learning trajectories sampled from policy π_θ to estimate the return expectation.

2) CRL in Dynamic Environments: Following the convention in [28], in this article, we consider the dynamic environment as an infinite sequence of stationary tasks where each task corresponds to the specific environmental dynamics within its time period and the same dynamics may recur more than once across different time periods. The time period is assumed to be long enough for the agent to get sufficient experience samples to finish policy learning for the associated task. Suppose that there is a space of MDPs denoted as $\mathcal{M}$, and a dynamic environment $\mathcal{D}$ changing over time in $\mathcal{M}$. The CRL agent interacts with $\mathcal{D} = [M_1, M_2, \ldots, M_{t-1}, M_t, \ldots]$, where each $M_t \in \mathcal{M}$ is a specific MDP/task that is stationary in the tth time period. In practice, the time period t can be thought of as a task inference period, where M_t is the corresponding task within it. Compared with M_{t-1}, M_t is either the same task as M_{t-1} or changes into another different new or previous one. In this process, the identity and actual change point of each task are both *unknown* to the agent.

It should be highlighted that to represent the policies for multiple tasks with a single model, the inputs (observations) must at the very least implicitly contain the discriminative information of the tasks. In this setting, the goal of the CRL agent in dynamic environments in time period t is to extend the acquired knowledge, accumulated from previously learned tasks $[M_1, M_2, \ldots, M_{t-1}]$, to the current task M_t, to learn a single policy to achieve the maximum return on all learned tasks $[M_1, M_2, \ldots, M_t]$

$$\theta_t^* = \arg\max_\theta \sum_{i=1}^{t} J_{M_i}(\theta) \tag{3}$$

where J_{M_i} is the expected return on task M_i. Note that the agent is expected to learn a sequence of tasks one by one strategically so that it can retain the previously acquired knowledge when learning new tasks. In other words, the policy π with parameter θ_t^* in (3) is an approximate optimal policy over all learned tasks $[M_1, M_2, \ldots, M_t]$. In this article, the overall performance of the learned policy across all tasks is the primary metric for judging the performance of CRL agents.

B. Chinese Restaurant Process

The term CRP [23] arises from an analogy of seating a sequence of customers in a Chinese restaurant with an infinite number of tables (i.e., clusters) and each table has infinite capacity. Each customer sits randomly at an occupied table with a probability proportional to the number of current customers at that table or an unoccupied table with a probability proportional to a hyperparameter α ($\alpha > 0$). The conditional probability for the tth customer sitting at the kth table is

$$p(z_t = k | z_{1:t-1}^*, \alpha) = \begin{cases} \dfrac{m_k^{(t-1)}}{t - 1 + \alpha}, & k \le K_{t-1} \\ \dfrac{\alpha}{t - 1 + \alpha}, & k = K_{t-1} + 1 \end{cases} \tag{4}$$

Authorized licensed use limited to: Bo Yuan. Downloaded on June 08,2023 at 13:42:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

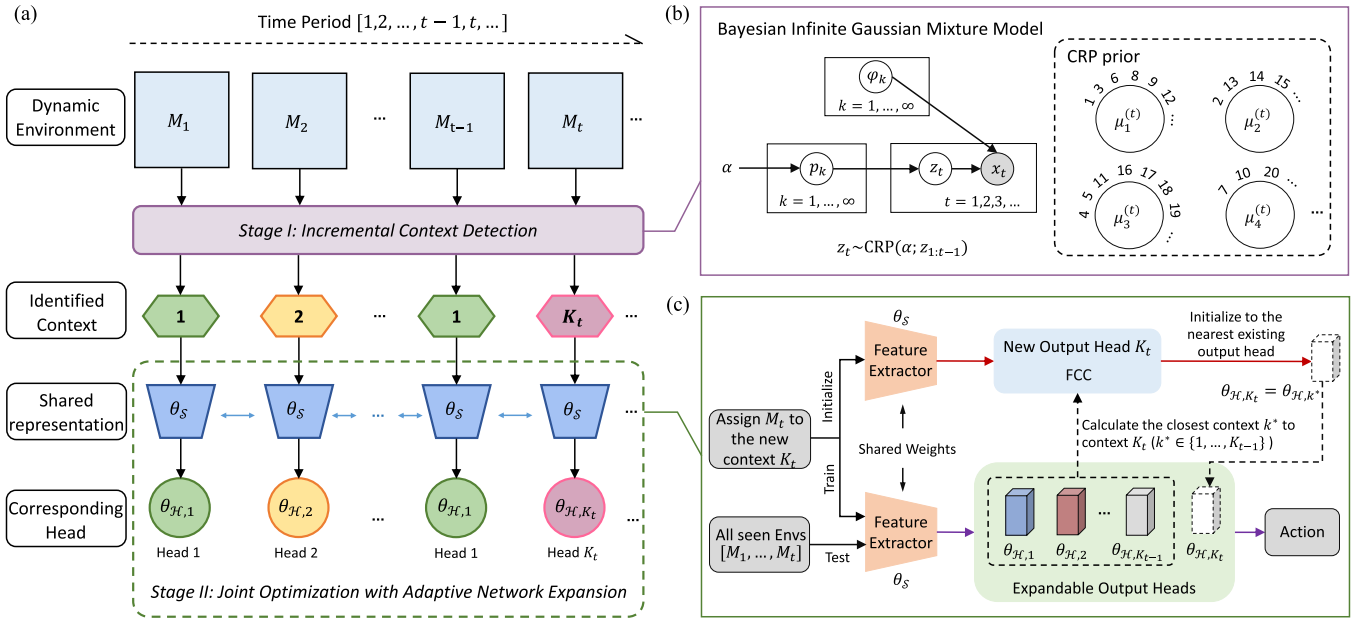


Fig. 2. Overview of DaCoRL in dynamic environments. (a) General framework of DaCoRL. The dynamic environment is represented by a sequence of stationary tasks $[M_1, M_2, \ldots, M_{t-1}, M_t, \ldots]$ with which the CRL agent interacts sequentially. The incremental context detection module either associates an existing context with the current task (e.g., M_{t-1} belongs to context 1) or instantiates a new context as needed (e.g., M_t belongs to a new context K_t) online. When training sequentially on different tasks, a policy network with a shared feature extractor (blue) and a set of expandable output heads corresponding to different contexts is maintained. (b) Bayesian IGMM with the CRP prior for context inference. (c) Multihead neural network expansion. For task M_t assigned to the new context K_t, we add a new output head and initialize it to the nearest existing output head and then train the whole network on M_t while keeping the performance on learned tasks unchanged.

where z_t is the latent cluster label of the tth customer; k is the table label; $z^*_{1:t-1} = \{z^*_1, z^*_2, \ldots, z^*_{t-1}\}$ represents the table labels assigned to $t-1$ customers; $m_k^{(t-1)}$ counts the number of assignments to table k up to time $t-1$; K_{t-1} denotes the number of occupied tables after the $(t-1)$th customer is seated; and α serves as the concentration parameter that controls the likelihood of new table.

The CRP can be viewed as equivalent to an IGMM under the assumption that the samples of a CRP are exchangeable.[3] In this article, we employ an IGMM to formulate the distribution of the dynamic environment, that is, we assume that the features of tasks in the dynamic environment present well-defined Gaussian-like distributions, which is the most common data pattern in practical applications, especially in high-dimensional spaces. Since tasks with different dynamics appear completely randomly in our problem settings, the prior distribution over these cluster structures can be well described via the CRP, which can provide a flexible structure in which the number of clusters is determined by the observed dynamic environments, and allow IGMM to predict the number of clusters while simultaneously performing model inference.

IV. Proposed Method

In this section, we give a detailed description of DaCoRL whose overview is shown in Fig. 2. We first introduce the key components including incremental context detection and adaptive network expansion. Then, we present a joint optimization scheme combining the proposed techniques with the canonical policy-based RL to achieve competent CRL in dynamic environments. For the sake of clarity, we here present an instantiation of DaCoRL using the vanilla policy gradient RL algorithm REINFORCE [52].

A. Incremental Context Detection

In dynamic environments, it is important to automatically detect and identify the changes in the environment, which can enable specialized policy learning for different tasks, alleviating catastrophic forgetting. In this article, to characterize the environmental dynamics, we inherit and extend the concept of "*context*" in [21] where a set of tasks with similar dynamics is regarded as the same context. Under this setting, the detection of environmental changes can be transformed into a context division procedure in the latent space.

However, compared with the single stationary environment investigated in [21], context division in dynamic environments brings significantly more challenges. In particular, the changes are usually infinite and highly uncertain and, without prior knowledge, it is hard to determine in advance the number of contexts that need to be instantiated.

To address this issue, we propose an incremental context detection module in DaCoRL, which can instantiate contexts when necessary in an incremental manner without any prior knowledge. Specifically, we formalize the task of context detection as an online Bayesian infinite Gaussian mixture clustering procedure. To do so, we define a set of environment features to represent the environmental dynamics and then perform clustering on the features by assuming a CRP prior distribution over the clustering structure. In CRP, the instantiation of new contexts is controlled by the concentration parameter α: the larger the value of α, the more contexts instantiated in general. On the one hand, when $\alpha = 0$, only a single context

[3]This means that, under any permutation of samples' ordering, the probability of a particular partitioning result is the same.

Authorized licensed use limited to: Bo Yuan. Downloaded on June 08,2023 at 13:42:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

is instantiated during the learning process. On the other hand, when $\alpha \to \infty$, a new context is instantiated for each task encountered. The complete procedure of incremental context detection with CRP is described in Algorithm 1, which mostly entails the next two steps.

1) Environment Feature Construction (Lines 1–2): Considering the discriminative task information is implicitly contained in the observation space, we can detect the changes in environmental dynamics by tracking the variation of the observation distribution directly. In practical implementations, we construct the features for a specific task from the collected observations. Specifically, before the start of RL training in each time period, the agent is required to explore the current environment using a random policy

$$\pi_r(a|s) = \text{Uniform}(A(s)) \tag{5}$$

where $A(s)$ is the set of available actions in the state s and Uniform$(\cdot)$ is the uniform distribution function. Subsequently, we construct the feature vector (x_t) for M_t so that x_t can represent the dynamics of the current environment from the observations $\mathcal{T}_\varepsilon = \{s_0^i, s_1^i, s_2^i, \ldots\}_{i=1}^m$, where m is the number of trajectories and i denotes the trajectory index.

Here, we approximate the set of collected observations as a Gaussian distribution and use its mean vector as the feature of the current environment. Naturally, we can obtain x_t directly from the original observation space

$$x_t = \frac{1}{N}\sum_{i=1}^{N} s_i \tag{6}$$

for the vector inputs, or from the embedding space

$$x_t = \frac{1}{N}\sum_{i=1}^{N} \phi(s_i) \tag{7}$$

for the visual inputs, where N is the number of states contained in $\mathcal{T}_\varepsilon$ and $\phi(s)$ can be represented by a random encoder [59] or a standard GAN model [60].

2) Context Detection via Online Incremental Clustering (Lines 3–12): To enable incremental context detection in dynamic environments, based on the constructed environment features, we employ online Bayesian inference to update the context models in a fully online fashion, avoiding the necessity for storing previously seen samples. The key step is to estimate the context parameters $\{\varphi_k\}_{k=1}^{K_t}$, which can build up a mapping between specific tasks and context variables in the latent space. In our implementations, the context model (predictive likelihood function) represents environment features using diagonal Gaussian distributions

$$p(x_t|\varphi_k) = \mathcal{N}(x_t; \mu_k, \sigma^2) \tag{8}$$

where μ_k is the mean of the Gaussian used to denote context k and σ^2 is a constant indicating the variance. Under this setting, the context centroids are just the mean vectors of the Gaussian distributions to be estimated: $\varphi_k = \{\mu_k\}$.

For a sequence of tasks $[M_1, M_2, \ldots, M_{t-1}, M_t]$, the first task is assigned to the first context by default. For M_t in the tth time period, we instantiate a new potential context $K_{t-1}+1$ and

Algorithm 1 Incremental Context Detection With CRP

Input: The learning task M_t in the tth time period;
The parameter set $\{\varphi_k^{(t-1)}\}_{k=1}^{K_{t-1}}$ of contexts already instantiated before the tth time period.
Parameter: Concentration parameter α.
Output: Task-to-context assignment z_t^* for M_t.
The parameters $\{\varphi_k^{(t)}\}_{k=1}^{K_t}$ of instantiated contexts.

1: Sample m trajectories from M_t using a uniform policy π_r: $\mathcal{T}_\varepsilon = \{\tau_i\}_{i=1}^m$, $\tau_i \sim \pi_r$.
2: Construct the feature vector x_t of M_t from $\mathcal{T}_\varepsilon$.
3: Initialize $\varphi_{K_{t-1}+1}^{(t-1)}$ of a new potential context as x_t.
4: Compute CRP prior $p(z_t = k|z_{1:t-1}^*, \alpha)$ for x_t.
5: Compute posterior probabilities of task-to-context assignment $p(z_t = k|z_{1:t-1}^*, x_t)$.
6: **if** $p(z_t = K_{t-1}+1|z_{1:t-1}^*, x_t) > p(z_t = k|z_{1:t-1}^*, x_t), \forall k \le K_{t-1}$ **then**
7: $K_t = K_{t-1}+1$, add $\varphi_{K_{t-1}+1}^{(t-1)}$ to $\{\varphi_k^{(t-1)}\}_{k=1}^{K_{t-1}}$.
8: **else**
9: $K_t = K_{t-1}$.
10: **end if**
11: Update parameters $\{\varphi_k^{(t-1)}\}_{k=1}^{K_t}$.
12: Calculate $z_t^* = \arg\max_k p(x_t|z_t = k, \varphi_k^{(t)}), \forall k \le K_t$ to obtain the final assignment of M_t.
13: **return** z_t^*, $\{\varphi_k^{(t)}\}_{k=1}^{K_t}$.

initialize the new context model parameter $\varphi_{K_{t-1}+1}^{(t-1)}$ as the feature vector x_t, regardless of whether M_t has been encountered before (line 3). After that, the posterior probabilities of the task-to-context assignment over all $K_{t-1}+1$ contexts are estimated (lines 4–5) by

$$\begin{aligned} &p\left(z_t = k|z_{1:t-1}^*, x_t, \varphi_k^{(t-1)}, \alpha\right) \\ &\quad \propto p\left(x_t|\varphi_k^{(t-1)}\right) p\left(z_t = k|z_{1:t-1}^*, \alpha\right). \end{aligned} \tag{9}$$

By combining the definition of the predictive likelihood in (8) and the CRP prior distribution in (4), the posterior distribution can be rewritten as

$$\begin{aligned} &p\left(z_t = k|z_{1:t-1}^*, x_t, \varphi_k^{(t-1)}, \alpha\right) \\ &\quad \propto \begin{cases} m_{k,t-1}\mathcal{N}\left(x_t; \mu_k, \sigma^2\right), & k \le K_{t-1} \\ \alpha\mathcal{N}\left(x_t; \mu_k, \sigma^2\right), & k = K_{t-1}+1 \end{cases} \end{aligned} \tag{10}$$

which is abbreviated to $p(z_t = k|z_{1:t-1}^*, x_t)$ in subsequent derivations for simplicity.

With the estimated $p(z_t = k|z_{1:t-1}^*, x_t)$ for each context, we can determine whether to keep the new context for M_t (lines 6–10). If the posterior probability of the potential context is greater than those of the K_{t-1} existing contexts, this new context is instantiated for M_t.

Next, we employ the EM procedure to update the context parameters in a fully incremental manner. Based on the inferred posterior probabilities, we update context parameters by optimizing the expected log-likelihood

$$\mathcal{L}\left(x_t|\varphi_k^{(t-1)}\right) = \mathbb{E}_{M_t} \log p\left(x_t|z_t = k, \varphi_k^{(t-1)}\right) \tag{11}$$

where $M_t \sim p(z_t = k|z_{1:t-1}^*, x_t)$. Suppose that all context models with the prior parameters $\varphi_k^{(t-1)}$ have been optimized

Authorized licensed use limited to: Bo Yuan. Downloaded on June 08,2023 at 13:42:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

up to the $(t-1)$th time period. The estimation of $\{\varphi_k^{(t)}\}_{k=1}^{K_t}$ can be updated based on the gradient (line 11)

$$\varphi_k^{(t)} \leftarrow \varphi_k^{(t-1)} + \eta_{k,t} \nabla_{\varphi_k^{(t-1)}} \mathcal{L}\left(x_t | \varphi_k^{(t-1)}\right) \quad \forall k \leq K_t \tag{12}$$

where $\eta_{k,t}$ is the learning rate[4] for the kth context in time period t. The gradient term $\mathcal{L}(x_t|\varphi_k^{(t-1)})$ can be derived as $p(z_t = k|z_{1:t-1}^*, x_t)\nabla_{\varphi_k} \log p(x_t|z_t = k, \varphi_k^{(t-1)})$.

Based on the updated context parameters, the identity z_t^* of M_t can be finally obtained by computing a MAP estimate on the predictive likelihood (line 12), selecting the context model that best fits the current environment.

B. Adaptive Network Expansion

To minimize the interference among contexts in CRL, we opt for an expandable neural network in DaCoRL, in which an output head is added synchronously with the newly instantiated context. Each output head specializes in a specific context, and the representation layers are shared among different contexts. This adaptively expandable network structure can parameterize a dedicated policy for each context without any unnecessary redundancy. In Fig. 2(a), the set of neural network weights is denoted by $\theta = \{\theta_{\mathcal{S}}, \theta_{\mathcal{H},1}, \theta_{\mathcal{H},2}, \dots\}$, where $\theta_{\mathcal{S}}$ is a set of weights for shared representation layers and $\theta_{\mathcal{H},k}$, $k \in \{1, 2, \dots\}$ are the parameters of the kth output head.

The policy network is initialized as a canonical single-head neural network for the forthcoming learning in the first context. When a new context is instantiated, the neural network is adaptively expanded by adding an output head whose structure is consistent with that of the existing output heads. For the newly added output head K_t, we propose the following three practical implementations for parameter initialization.

1) *Random:* The weights of the newly added output head are initialized to random values. In this case, the policy of the newly instantiated context inherits the representation module of the learned policies, while its output layer is trained from scratch.
2) *Random-previous Head:* It randomly selects one of the trained output heads and then initializes the weights of the newly added output head to those of the selected one, that is, $\theta_{\mathcal{H},K_t} = \theta_{\mathcal{H},k}$, where $k = Uniform(\{1, 2, \dots, K_{t-1}\})$. This method enables the new policy to inherit knowledge from the learned context. However, it is likely that the cloned policy may hinder the CRL agent's ability to properly explore the task space corresponding to the current context, especially when there are significant differences between the two contexts.
3) *Nearest-Previous Head:* It initializes the newly added output head to a specific trained one whose associated context is nearest to the current instantiated context in the latent context space, that is, $\theta_{\mathcal{H},K_t} = \theta_{\mathcal{H},k^*}$, where $k^* = \arg\min_k dist(\varphi_k, \varphi_{K_t})$, $k \in \{1, 2, \dots, K_{t-1}\}$ and $dist(\varphi_k, \varphi_{K_t})$ denotes the distance between contexts k and K_t in the latent context space.

Intuitively, the third implementation is most likely to encourage forward transfer. The experimental results and analysis in Section V-F provide further elaboration on this point.

C. Joint Optimization Scheme

In the policy optimization stage, we integrate the above components with policy-based RL algorithms to achieve efficient CRL in dynamic environments. Furthermore, we employ the knowledge distillation technique [21] to mitigate the catastrophic interference caused by distribution drifts among contexts as well as the minor interference due to the differences among tasks within the same context. Taking REINFORCE as the underlying policy-based RL algorithm as an example, the optimization objective of DaCoRL is derived as follows.

First of all, we rewrite the original loss function of REINFORCE in (1) with the context label variable z_t^* as

$$\mathcal{L}_{\text{ori}}(\theta_{z_t^*}) = \mathbb{E}_{\tau \sim \pi_{\theta_{z_t^*}}(\tau)}\left[R(\tau)\right] \tag{13}$$

where $\theta_{z_t^*} = \{\theta_{\mathcal{S}}, \theta_{\mathcal{H},z_t^*}\}$ and $\pi_{\theta_{z_t^*}}$ is the policy corresponding to the context associated with the current task M_t.

For possible interference among tasks, we adopt the knowledge distillation technique to construct a regularization term in the probability distribution estimation of actions, to preserve the previously learned policies. Specifically, we regard the policy network from the last time period as the teacher network, expressed as π_{θ^-}, and the current policy network to be trained as the student network, denoted by π_θ. Then, Kullback–Leibler divergence is used to constrain the difference between these two networks' policies. Thus, the distillation loss of the output head for context k is defined as

$$\mathcal{L}_{\mathcal{D}_k}(\theta_k) = \mathbb{E}_{\tau \sim \pi_{\theta_{z_t^*}}(\tau)}\left[KL\left[\pi_{\theta_k}(\cdot|s), \pi_{\theta_k^-}(\cdot|s)\right]\right] \tag{14}$$

where $\theta_k = \{\theta_{\mathcal{S}}, \theta_{\mathcal{H},k}\}$. In the tth time period, considering all K_t output heads in the policy network, the distillation loss term sums up as

$$\mathcal{L}_{\mathcal{D}}(\theta) = \sum_{k=1}^{K_t} \mathcal{L}_{\mathcal{D}_k}(\theta_k) \tag{15}$$

where $\theta = \{\theta_{\mathcal{S}}, \{\theta_{\mathcal{H},k}\}_{k=1}^{K_t}\}$ denotes the set of all weights of the policy network.

Finally, to optimize a policy network that can guide the agent to make proper decisions in dynamic environments without being adversely affected by catastrophic forgetting, we combine (13) and (15) to form a joint optimization scheme. Namely, we solve the CRL problem in dynamic environments by the following optimization objective:

$$\begin{aligned} &\max_{\theta_{\mathcal{S}}, \theta_{\mathcal{H}}} \mathcal{L}_{\text{ori}}(\theta_{z_t^*}) - \lambda \mathcal{L}_{\mathcal{D}}(\theta) \\ &\theta_{z_t^*} = \{\theta_{\mathcal{S}}, \theta_{\mathcal{H},z_t^*}\} \\ &\theta = \{\theta_{\mathcal{S}}, \{\theta_{\mathcal{H},k}\}_{k=1}^{K_t}\} \end{aligned} \tag{16}$$

[4]Inspired by sequential K-Means clustering adopted in [21], the updating amplitude of cluster center is inversely proportional to the number of samples contained in the cluster. In our implementations, we set $\eta_{k,t}$ to $1/(m_{k,t-1} + p(z_t = k|z_{1:t-1}^*, x_t))$, where $m_{k,t-1}$ is the number of tasks assigned to context k up to the $(t-1)$th time period and $p(z_t = k|z_{1:t-1}^*, x_t)$ is the posterior probability that x_t is assigned to cluster k.

Authorized licensed use limited to: Bo Yuan. Downloaded on June 08,2023 at 13:42:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Algorithm 2 DaCoRL

Input: Dynamic environment $\mathcal{D} = [M_1, M_2, \ldots, M_T]$. Single-head policy network π_θ with random weights $\theta = \{\theta_{\mathcal{S}}^{(0)}, \theta_{\mathcal{H},1}^{(0)}\}$.

Parameter: Concentration parameter α; Learning rate β; Distillation regularization coefficient λ.

Output: Parameter set $\{\varphi_k\}_{k=1}^{K_T}$ of instantiated contexts; Approximate optimal policy parameters θ^* for $\mathcal{D}$.

1: **for** each time period $t \in \{1, 2, \ldots, T\}$ **do**
2: **if** $t = 1$ **then**
3: Sample randomly and construct the feature vector x_1 for the task M_1.
4: Instantiate M_1 as the first context: $\varphi_1^{(1)} = x_1$, $z_1^* = 1$.
5: Set $K_1 = 1$, $m_1^{(1)} = 1$ in the CRP model.
6: Update the policy network from scratch using the canonical policy gradient method to obtain θ^*:
$$\theta \leftarrow \theta + \beta \nabla_\theta \mathcal{L}_{ori}.$$
7: **else**
8: Infer the task-to-context assignment z_t^* for M_t using incremental context detection with CRP:
$z_t^*, \{\varphi_k^{(t)}\}_{k=1}^{K_t} \leftarrow$ Algorithm 1$(M_t, \{\varphi_k^{(t-1)}\}_{k=1}^{K_{t-1}}, \alpha)$.
9: Update the CRP model according to z_t^*:
$$m_k^{(t)} = m_k^{(t-1)}, \forall k \leq K_t; \;\; m_{z_t^*}^{(t)} = m_{z_t^*}^{(t-1)} + 1.$$
10: **if** $z_t^* = K_{t-1} + 1$ **then**
11: Expand π_θ by adding an output head initialized with the nearest-previous head, add $\theta_{\mathcal{H},K_t}$ to θ.
12: **end if**
13: Set $\pi_{\theta^-} = \pi_\theta$ for distillation.
14: Update the policy network using (16) to obtain θ^* :
$$\theta \leftarrow \theta + \beta \nabla_\theta (\mathcal{L}_{ori} - \lambda \mathcal{L}_{\mathcal{D}}).$$
15: **end if**
16: **end for**
17: **return** $\{\varphi_k\}_{k=1}^{K_T}$, θ^*.

where $\lambda \in [0, 1]$ is a coefficient to control the tradeoff between learning the new policy and preserving the learned policies.

The complete procedure of DaCoRL is summarized in Algorithm 2 where the agent interacts with a dynamic environment $\mathcal{D} = [M_1, M_2, \ldots, M_T]$. In the first time period $t = 1$, the task M_1 is instantiated as the first context with parameter $\varphi_1^{(1)}$ (line 4). We initialize the CRP model with $K_1 = 1$, $m_1^{(1)} = 1$ (line 5) and train the single-head policy network from scratch using the canonical policy gradient method (line 6). In the tth ($t \geq 2$) time period, we first apply the incremental context detection module to identify the context to which the current task belongs (line 8) and update the CRP model based on the identity of M_t for future use (line 9). Then, we employ an expandable multihead neural network for policy optimization. Specifically, when a new context is instantiated, the policy network is synchronously expanded by adding an output head initialized with the nearest-previous head (lines 10–12). Next, the knowledge distillation regularization term is integrated into the loss of the original RL algorithm to reduce the catastrophic forgetting of learned tasks, and the parameters of the policy network are updated till convergence (lines 13–14). Finally, K_T contexts with parameters $\{\varphi_k\}_{k=1}^{K_T}$ and the optimal policy π_{θ^*} are obtained for all learned tasks.

V. Experiments and Evaluations

In this section, we conduct comprehensive experiments on several continuous control tasks from robot navigation to MuJoCo locomotion to demonstrate the effectiveness of our method. We design a variety of sequential learning tasks with diverse changes in the underlying dynamics. These problem settings are expected to be representative of the dynamic environments that RL agents may encounter in real-world scenarios. The following are the overarching questions that we aim to answer from our experiments and analysis.

- Q1 Does DaCoRL successfully achieve better CRL in various dynamic environments compared with existing methods?
- Q2 How does the initialization strategy of the newly added output head affect the performance of DaCoRL?
- Q3 How does the number of instantiated contexts in the latent space affect the performance of DaCoRL?
- Q4 Can DaCoRL achieve a positive forward transfer during the learning process?
- Q5 How is DaCoRL's generalization ability to previously unseen tasks?

A. Datasets

1) Robot Navigation [27], [28]: It contains three types of dynamic environments with parametric variation across tasks. Types I, II, and III indicate that the dynamic environments are created in terms of the goal position (changes in the reward function), the puddles positions (changes in the state transition function), and both the goal and puddles positions (changes in both above functions), respectively. In each task, the agent needs to move to a goal position within a unit square. The state consists of the agent's current 2-D position, and the action corresponds to the 2-D velocity commands in the range of $[-0.1, 0.1]$. The reward is equal to the negative squared distance to the goal position minus a small control cost that is proportional to the action's scale. Each learning episode always starts from a given position and terminates when the agent is within 0.01 from the goal or when the episode length is greater than 100. We choose these commonly used domains as they are well-understood, suitable for highlighting the mechanism and verifying the effectiveness of our method in a straightforward manner.

2) MuJoCo Locomotion [28]: It contains three locomotion tasks with parametric variation and growing dimensions of state-action spaces. These tasks require a one-legged hopper, a planar cheetah, or a 3-D quadruped ant robot to run at a particular velocity along 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 preset target velocity. The dynamic environment is designed to apply parametric variations in the target velocity within a 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

Authorized licensed use limited to: Bo Yuan. Downloaded on June 08,2023 at 13:42:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

the agent falls down or when the episode length is greater than 100. We choose these domains to further evaluate the efficiency of our method on more sophisticated domains.

In this article, we follow the multitask learning benchmark construction described in [61] and extend the original observation space for each task by adding the associated parametric variations, to make the observation input discriminative among tasks and make it sensible to represent policies for sequential tasks with a single model.[5] Without loss of generality, we here insert the parametric variations[6] into random dimensions of the initial observation space. In our experiments, we uniformly generate four Gaussian clusters (i.e., expected contexts) for each dynamic environment in its parametric variation space. Each of the first three clusters consists of 12 tasks and the fourth cluster contains 14 tasks. We sequentially arrange these $T = 50$ tasks in a random order, resulting in a dynamic environment $\mathcal{D} = [M_1, M_2, \ldots, M_T]$. For DaCoRL (Oracle), the supervised version of DaCoRL mentioned in our experiments, we make available the task-to-context assignments based on the results of cluster generation. By contrast, the correspondence between tasks and contexts is unknown and needs to be identified by DaCoRL itself.

B. Baselines

We evaluate our method in comparison to the following four state-of-the-art baseline methods and one supervised version of our proposed DaCoRL in dynamic environments.

1) *Naive:* It refers to canonical RL methods (e.g., REINFORCE [52], PPO [62], [63]) that simply train a policy model during the learning process, without paying attention to any possible environmental changes or forgetting.
2) *CRLUnsup [56]:* It detects environmental changes by observing the ability of the agent to perform the task. When the changes are detected, it triggers the memory consolidation procedure employing EWC to preserve previously learned policies from catastrophic forgetting.
3) *CDKD:* It is adapted from the IQ [21] framework that can alleviate catastrophic interference for value-based RL in stationary environments. In this article, we extend IQ to policy-based RL with context division and knowledge distillation (renamed as CDKD) and set the true number of contexts in advance so that it can conduct continual learning in dynamic environments.
4) *LLIRL [28]:* It is a recent method that focuses on fast adaptation in nonstationary environments, which maintains a library that contains an infinite mixture of parameterized environment models for task clustering and optimizes the learning parameters based on the knowledge accumulated by the previously learned similar tasks to maximize return in the current environment, *without* considering the forgetting of previously learned policies.
5) *DaCoRL (Oracle):* This approach can be regarded as a supervised version of DaCoRL, in which the agent is informed in each time period of the specific task-to-context mapping (i.e., context identities are made available).

To handle continuous control tasks, we use policy search with nonlinear function approximation in our experiments. For the navigation tasks, we perform gradient updates using the vanilla policy gradient RL algorithm REINFORCE. In addition, we employ PPO as the base RL algorithm to learn the more challenging MuJoCo locomotion tasks.

C. Implementation

We adopt a similar network architecture for all tasks. The policy of DaCoRL is approximated by a feed-forward neural network that contains a fully connected hidden layer (with 200 units) used as the feature extractor and an expandable output head module used as the action distribution predictor, where each output head consists of a 200-unit fully connected hidden layer and a fully connected output layer. The hidden layers are connected by ReLU nonlinearity, following the network configuration for these tasks in [28]. For a fair comparison, the network architecture of CDKD is set to the same as that of DaCoRL and the number of contexts predetermined for CDKD is also set to be consistent with that automatically detected by DaCoRL. For Naive, CRLUnsup, and LLIRL, each policy network consists of two 200-unit hidden layers connected by ReLU nonlinearity and a fully connected output layer.

During the learning process, we train the model for 1k policy iterations on each task and evaluate the policy performance by testing the current policy on all tasks in the dynamic environment every 100 iterations. All results reported are the average performance over five independent runs.

D. Evaluation Metrics

Following the convention in [27], [28], we define two metrics to systematically evaluate our method. The first one is the average return over a batch of test episodes on all tasks in the dynamic environment, which is used to evaluate the overall performance of the model on all tasks after a fixed number of policy iterations during training in real time. It is defined as

$$\mathcal{R}_{\text{ave}} = \frac{1}{Tm} \sum_{i=1}^{T} \sum_{j=1}^{m} R(\tau_{ij}) \tag{17}$$

where T is the number of tasks in the dynamic environment; m is the number of episodes tested for each task; and $R(\tau_{ij})$ is the cumulative reward obtained in the jth test episode of task i. The other is the average return over all test episodes, which evaluates the average performance of the model over the entire training process. It is defined as

$$\bar{\mathcal{R}}_{\text{ave}} = \frac{1}{J} \sum_{i=1}^{J} \mathcal{R}_{\text{ave}}^{(i)} \tag{18}$$

where J is the number of $\mathcal{R}_{\text{ave}}$ evaluations in the learning process, and $\mathcal{R}_{\text{ave}}^{(i)}$ is $\mathcal{R}_{\text{ave}}$ in the ith evaluation.

[5]This is a prerequisite for effective multitask learning with a single model. Serious conflicts in model training will occur when the same observation input corresponds to tasks with different objectives.

[6]The positions of goal/puddles/both of them for Type I/Type II/Type III of navigation tasks and the target velocity for MuJoCo locomotion tasks.

Authorized licensed use limited to: Bo Yuan. Downloaded on June 08,2023 at 13:42:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

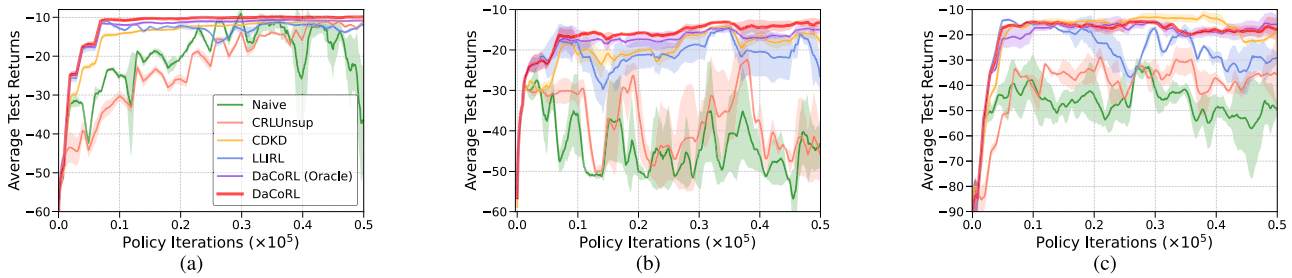


Fig. 3. Testing curves of the average returns over all tasks in the dynamic environments ($\mathcal{R}_{\text{ave}}$) of the navigation tasks. Here and in related figures in the following, K_T is the number of instantiated contexts by DaCoRL and the solid lines and shaded regions denote the means and standard deviations of average test returns, respectively, across five runs. (a) Type I, $K_T = 5$. (b) Type II, $K_T = 6$. (c) Type III, $K_T = 4$.

TABLE I

NUMERICAL RESULTS OF $\bar{\mathcal{R}}_{\text{AVE}}$ OF ALL METHODS IN THE NAVIGATION TASKS (BASED ON THE RESULTS IN FIG. 3. HERE AND IN RELATED TABLES, THE CONFIDENCE INTERVALS ARE STANDARD DEVIATIONS. THE BEST PERFORMANCE IS MARKED IN BOLDFACE.)

Task	Type I	Type II	Type III
Naive	-20.44 ± 1.07	-43.55 ± 2.25	-47.28 ± 3.98
CRLUnsup	-22.43 ± 0.97	-37.73 ± 1.82	-40.32 ± 1.02
CDKD	-14.34 ± 0.30	-19.86 ± 0.69	$\mathbf{-19.13 \pm 0.95}$
LLIRL	-14.24 ± 0.24	-21.08 ± 1.92	-25.92 ± 0.41
DaCoRL	$\mathbf{-11.93 \pm 0.43}$	$\mathbf{-16.15 \pm 0.30}$	-19.58 ± 0.08
DaCoRL (Oracle)	-12.79 ± 0.26	-17.61 ± 0.03	-20.46 ± 1.19

Remark: For task M, the test procedure of DaCoRL is conducted in two steps. 1) Policy selection. It first constructs the feature vector x following the procedure in lines 1–2 of Algorithm 1 and then determines the identity z^* by computing a MAP estimate on the predictive likelihood of K_T contexts. The policy corresponding to the output head z^* is the final policy selected for M. 2) Policy execution. The agent applies the selected policy on M to evaluate its performance in terms of the cumulative reward in each episode.

E. Results

To investigate the effectiveness of our proposed method (Q1), we present the results of DaCoRL and all baselines in three types of navigation tasks. Fig. 3 shows the average episodic returns over all tasks during training according to (17), and Table I reports the numerical results in terms of the average returns over $1000/100 * 50$ tests throughout the whole training process according to (18). For DaCoRL, the numbers of instantiated contexts are $K_T = 5$, $K_T = 6$, and $K_T = 4$ for the three types of navigation tasks, respectively. In Fig. 3, it is clear that DaCoRL is significantly superior to all baselines in terms of the average test return and stability. Naive shows the worst forgetting and performance fluctuations since it does not employ any mechanism to overcome data drifts in dynamic environments. CRLUnsup is slightly better than Naive due to the use of the EWC technique. Its performance, however, is still far from that of other baselines due to the weak capacity to identify environmental changes just by looking at the cumulative reward curve. CDKD and LLIRL perform clearly better than the above two baselines. Nevertheless, the fixed policy network structure may lead to an over-constrained optimization objective at the early stage of CDKD training, thus limiting the learning progress. Although LLIRL employs separate neural networks to learn tasks from different contexts, which completely hinders the interference among between-context tasks, it involves more weight updating and model storage of neural networks and is primarily intended for faster adaptation as opposed to forgetting prevention. At the same time, it does not include any consideration of interference among tasks within the same context. As shown in Fig. 4, we visualize the average performance degradation (called *forgetting*) of LLIRL and DaCoRL after ending each task's training and all sequential tasks' training in the navigation tasks. It is clear to see that compared with DaCoRL (train a multihead network using knowledge distillation technique), LLIRL training multiple separate context-specific neural networks has a substantially higher degree of forgetting of the learned task policies.

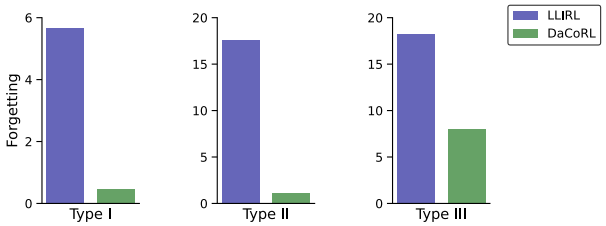


Fig. 4. Average forgetting of LLIRL versus DaCoRL in the navigation tasks. For each task i, we set forgetting as the decrease of performance after ending its training, $F_i = \mathcal{R}_i(i) - \mathcal{R}_i(T)$, where $\mathcal{R}_i(j)$ denotes the test returns for task i after the jth time period training. Here, we report $F = {}^1\!/\!_T \sum_{i=1}^{T} F_i$. A positive value of F indicates that the agent forgets a certain degree of learned information during the whole learning process, while the negative value indicates backward transfer.

By contrast, due to the effective context division of dynamic environments in the latent space, along with an expandable multihead policy network and the knowledge distillation technique, DaCoRL can achieve competent continual learning of sequential tasks in dynamic environments. In particular, it is on a par with [see Fig. 3(c)] or even outperforms [see Fig. 3(a) and (b)], its supervised version [DaCoRL (Oracle)] where the task-to-context assignments are provided in advance. These results reveal that the incremental context detection module may produce more rational and valuable context division for continual learning than the predetermined task-to-context assignments. Furthermore, the results in Table I indicate that DaCoRL receives significantly larger or comparable average returns than all baselines during the entire learning process, and it also generally features smaller standard deviations in performance than the baselines.

Authorized licensed use limited to: Bo Yuan. Downloaded on June 08,2023 at 13:42:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

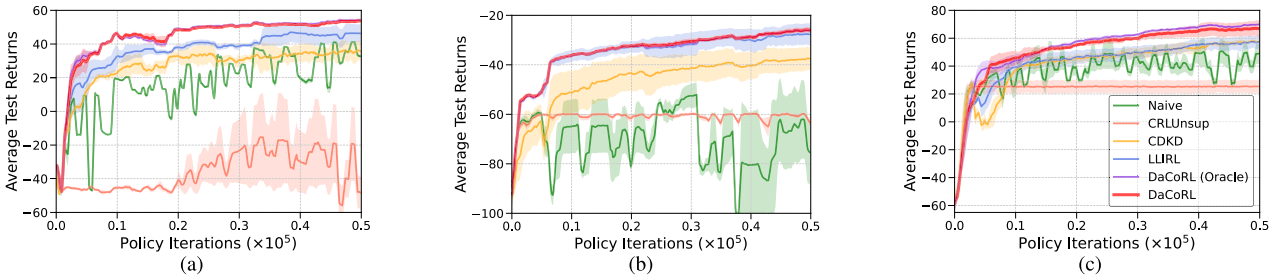


Fig. 5. Testing curves of the average returns over all tasks in the dynamic environments ($\mathcal{R}_{\text{ave}}$) of the MuJoCo locomotion tasks. $K_T = 4$ contexts are instantiated for DaCoRL in all tasks. (a) Hopper. (b) HalfCheetah. (c) Ant.

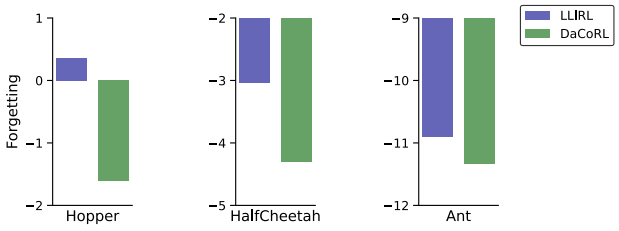


Fig. 6. Statistics of average forgetting of LLIRL versus DaCoRL in the MuJoCo locomotion tasks.

TABLE II

NUMERICAL RESULTS OF $\bar{\mathcal{R}}_{\text{AVE}}$ OF ALL METHODS IN THE MUJOCO LOCOMOTION TASKS (BASED ON THE RESULTS IN FIG. 5.)

Task	Hopper	HalfCheetah	Ant
Naive	16.18 ± 0.32	-69.98 ± 6.44	36.39 ± 5.87
CRLUnsup	-35.87 ± 9.38	-61.08 ± 0.55	23.37 ± 4.40
CDKD	26.47 ± 3.73	-46.36 ± 6.63	41.24 ± 1.55
LLIRL	35.36 ± 0.75	-34.84 ± 3.10	42.76 ± 4.25
DaCoRL	$\mathbf{44.07 \pm 0.46}$	$\mathbf{-34.11 \pm 0.62}$	$\mathbf{52.18 \pm 1.34}$
DaCoRL (Oracle)	44.57 ± 0.45	-34.31 ± 0.58	54.09 ± 1.87

To further demonstrate the scalability and flexibility of our method, we evaluate DaCoRL and all baselines on MuJoCo locomotion tasks. All results are summarized in Fig. 5 and Table II, from which we can see that DaCoRL consistently exhibits better and more stable performance than all baseline methods, even in these complex dynamic environments. More specifically, DaCoRL instantiates totally four contexts ($K_T = 4$) for each dynamic environment, which is consistent with the given contexts in DaCoRL (Oracle). Since the learning curves of these two methods are largely coincident in all tasks in Fig. 5, it further confirms that DaCoRL can accurately identify environmental context changes in a fully self-adaptive manner and can achieve comparable performance with the case where the task-to-context assignments are known in advance. In addition, we also visualize the forgetting features of DaCoRL and LLIRL in this domain in Fig. 6, which once again confirms the efficiency of DaCoRL in alleviating forgetting.

F. Analysis

1) Influence of the Initialization of Output Heads: To address Q2, we evaluate DaCoRL in the navigation tasks with different initialization strategies described in Section IV-B. The average returns over all tasks during the first 100 policy iterations are shown in Fig. 7. Here, we shorten the three

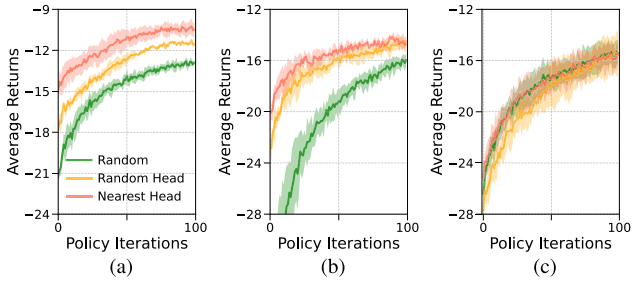


Fig. 7. Average training episodic return over all tasks per iteration of DaCoRL with different output head initialization implementations in the navigation tasks. (a) Type I. (b) Type II. (c) Type III.

initialization implementations to "Random," "Random Head," and "Nearest Head," respectively, to ensure readability.

From Fig. 7, it is clear that the initialization by Nearest Head can enable DaCoRL to attain a better initial policy and more positive forward transfer to the new task, which is consistent with our analysis in Section IV-B. Meanwhile, compared with Random initialization, Random Head exhibits obvious positive forward transfer in Type I and Type II environments and negative forward transfer in the Type III environment. Referring to [27], a possible explanation is that, in the Type III environment, Random Head often chooses an initial policy that hinders the exploration of the new task such that the agent needs to spend more time in the early training stage to counter the old policy, resulting in slow performance improvement.

2) Influence of the Number of Instantiated Contexts (K_T): Intuitively, instantiating a separate context for each task contained in the dynamic environment (i.e., $K_T = T$, task-specific output head within an individual time period) is likely to result in an optimal policy for the agent. However, because of the intricate network architecture and challenging model training procedure, it is problematic to use in practice, particularly when the dynamic environment comprises a large number of tasks. Thus, it is necessary to investigate the influence of the number of contexts on the performance of DaCoRL (Q3).

We change the numbers of instantiated contexts in DaCoRL by varying the concentration parameter α in the CRP in the navigation tasks. Since the environmental features are highly correlated with their variation parameters, that is, intuitively, taking the Type I environment, for example, the tasks with adjacent goal positions are more similar to each other and tend to belong to the same context. Hence, we use the goal position in the 2-D coordinate as a visualization to reveal the clustering patterns of contexts for the Type I environment. The results

Authorized licensed use limited to: Bo Yuan. Downloaded on June 08,2023 at 13:42:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

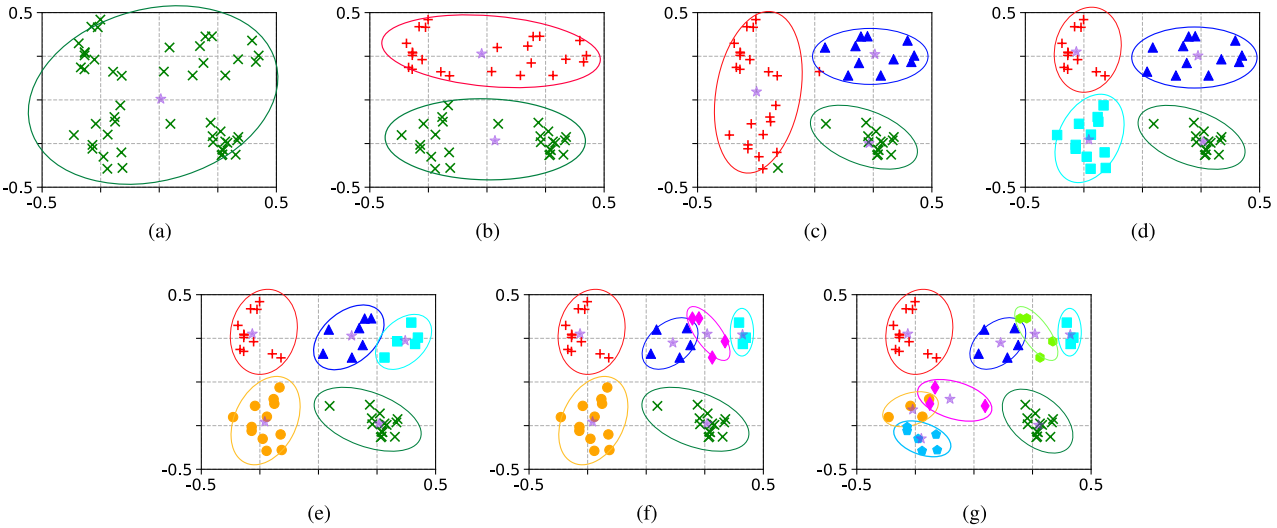


Fig. 8. Clustering patterns of contexts in DaCoRL with different numbers of instantiated contexts in the Type I navigation environment. (a) One context. (b) Two contexts. (c) Three contexts. (d) Four contexts. (e) Five contexts. (f) Six contexts. (g) Eight contexts.

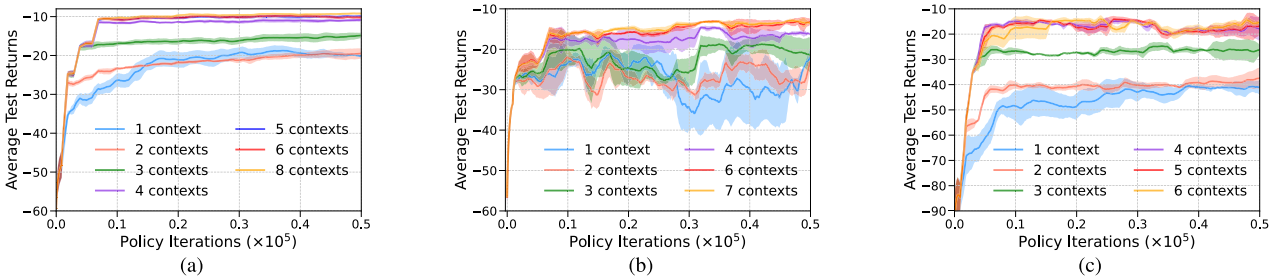


Fig. 9. Average returns over all tasks in the dynamic environments ($\mathcal{R}_{\text{ave}}$) of DaCoRL with different numbers of instantiated contexts in the navigation tasks. (a) Type I. (b) Type II. (c) Type III.

TABLE III

NUMERICAL RESULTS OF $\bar{\mathcal{R}}_{\text{AVE}}$ OF DACORL WITH DIFFERENT NUMBERS OF INSTANTIATED CONTEXTS IN THE ROBOT NAVIGATION TASKS (BASED ON THE RESULTS IN FIG. 9)

K_T	Type I	Type II	Type III
1	-22.86 ± 1.10	-27.34 ± 3.42	-47.16 ± 3.69
2	-22.32 ± 0.34	-26.80 ± 0.43	-42.08 ± 1.35
3	-17.09 ± 0.88	-22.59 ± 2.10	-28.74 ± 0.89
4	-12.89 ± 0.30	-18.00 ± 0.96	$\mathbf{-19.58 \pm 0.08}$
5	$\mathbf{-11.93 \pm 0.43}$	–	$\mathbf{-19.80 \pm 0.42}$
6	$\mathbf{-11.89 \pm 0.43}$	$\mathbf{-16.15 \pm 0.30}$	$\mathbf{-19.86 \pm 0.56}$
7	–	$\mathbf{-15.79 \pm 0.21}$	–
8	$\mathbf{-11.48 \pm 0.19}$	–	–

are shown in Fig. 8, where the star shapes represent the centroids of instantiated contexts obtained from the incremental context detection procedure, and other markers represent tasks (M_i, $i \in [1, 2, \ldots, T]$) in the dynamic environment. It is clear that our incremental context detection module can capture the appropriate context patterns under different concentration parameter settings in a fully online manner, regardless of how many contexts are ultimately instantiated.

Additionally, we show the performance of DaCoRL with different numbers of instantiated contexts in the three types of navigation environments in Fig. 9 and Table III according to (17) and (18), respectively. Overall, the results are consistent with the intuition that DaCoRL tends to achieve better performance with more contexts instantiated. Meanwhile, maintaining fewer contexts means that tasks with large differences may be clustered into the same context, leading to more interference during policy network training. Nevertheless, it should be noted that in Fig. 9 and Table III, the performance gains become relatively minor when the number of contexts exceeds a certain value. Consequently, for the sake of model complexity and training cost, we recommend choosing a moderate number of instantiated contexts by adjusting the value of α. For instance, in our experiments, $K_T = 5$ for Type I, $K_T = 6$ for Type II, and $K_T = 4$ for Type III are sufficient to obtain reasonable performance in these navigation tasks.

We also record the rough value range of the concentration parameter α related to the specific number of contexts. The results are presented in Appendix C-A, from which we empirically found that this hyperparameter is relatively insensitive, and all α values within the corresponding continuous interval can result in the same context instantiation results. This insensitivity ensures a certain degree of simplicity and convenience of parameter tuning in practical implementations.

3) Forward Transfer: In DaCoRL, tasks with similar dynamics are grouped into the same context and share the same policy throughout the CRL process. In each time period, DaCoRL retrieves the policy corresponding to the most similar context as the initial policy for the current learning process, which can enable the agent to get better startup performance on each new task. To validate this feature (Q4), we enumerate

Authorized licensed use limited to: Bo Yuan. Downloaded on June 08,2023 at 13:42:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE IV
NUMERICAL RESULTS IN TERMS OF THE AVERAGE INITIAL PERFORMANCE OVER ALL SEQUENTIAL TASKS DURING TRAINING IN THE ROBOT NAVIGATION AND MUJOCO LOCOMOTION TASKS

Task	Type I	Type II	Type III	Hopper	HalfCheetah	Ant
Naive	-22.17 ± 1.83	-43.63 ± 2.42	-50.86 ± 5.18	8.67 ± 0.47	-75.11 ± 4.57	30.64 ± 4.39
CRLUnsup	-23.07 ± 1.23	-39.09 ± 0.58	-44.78 ± 0.90	-39.40 ± 9.42	-63.55 ± 0.47	20.91 ± 4.56
CDKD	-20.11 ± 0.60	-31.34 ± 1.77	-25.96 ± 0.86	24.42 ± 3.93	-49.66 ± 7.64	34.57 ± 2.43
LLIRL	-19.08 ± 0.51	-23.80 ± 1.71	-33.47 ± 1.36	28.16 ± 0.77	-37.63 ± 3.55	40.53 ± 3.70
DaCoRL	$\mathbf{-14.92 \pm 0.74}$	$\mathbf{-20.11 \pm 0.78}$	$\mathbf{-25.14 \pm 0.67}$	$\mathbf{37.28 \pm 0.46}$	$\mathbf{-36.02 \pm 0.13}$	$\mathbf{49.07 \pm 1.38}$

TABLE V
NUMERICAL RESULTS IN TERMS OF THE AVERAGE TEST RETURN OVER 50 DIFFERENT TASKS THAT HAVE NOT BEEN SEEN DURING TRAINING IN ROBOT NAVIGATION AND MUJOCO LOCOMOTION TASKS

Task	Type I	Type II	Type III	Hopper	HalfCheetah	Ant
Naive	-33.78 ± 19.51	-39.82 ± 9.78	-50.19 ± 8.02	39.36 ± 0.90	-69.84 ± 12.17	55.26 ± 6.01
CRLUnsup	-11.29 ± 0.59	-42.52 ± 9.09	-38.24 ± 16.41	-43.41 ± 12.73	-62.92 ± 1.57	34.13 ± 4.34
CDKD	-11.86 ± 0.24	-22.46 ± 2.06	$\mathbf{-18.77 \pm 3.93}$	34.25 ± 9.38	-36.14 ± 4.05	63.26 ± 1.86
LLIRL	-14.52 ± 1.28	-39.84 ± 1.26	-35.94 ± 0.15	48.76 ± 3.39	-27.67 ± 3.85	59.39 ± 2.25
DaCoRL	$\mathbf{-11.28 \pm 0.22}$	$\mathbf{-17.58 \pm 0.89}$	-23.38 ± 4.46	$\mathbf{53.45 \pm 0.90}$	$\mathbf{-26.36 \pm 1.20}$	$\mathbf{67.65 \pm 4.24}$

the average training episodic return for each algorithm at the first policy iteration on each task, and the results are shown in Table IV. In comparison to all baselines, DaCoRL consistently achieves better initial startup performance in all experimental dynamic environments, displaying a strong positive forward transfer ability.

4) Generalization Results: To investigate the generalization performance of DaCoRL in unseen tasks (Q5), we randomly generate 50 different tasks that follow the same distribution as the training tasks but have not been seen during a training in each dynamic environment. Then, we test the policies learned by DaCoRL and all baselines in these tasks, where each policy is tested on a specific task for 100 times. The average test returns over all tasks are shown in Table V. In general, DaCoRL features superior generalization ability in most unseen tasks compared with all baselines, except on Type III navigation tasks, where its generalization performance is slightly inferior to CDKD but still significantly better than the other three baselines. This phenomenon is explainable since CDKD is the most similar algorithm to DaCoRL and is likely to be comparable to DaCoRL especially when the numbers of contexts in both methods are consistent.

5) Sample Efficiency: Compared with Naive (no consideration of environmental changes) and CRLUnsup (identifying environmental changes by evaluating the cumulative rewards during training), methods with a separate context detection phase (i.e., CDKD, LLIRL, and DaCoRL) are required to explore the current environment using a uniform policy and collect extra samples for context inference. We record the number of agent–environment interaction episodes during training on sequential tasks for all approaches, and the results are summarized in Appendix C-B. Combined with the experimental results shown in the preceding part of this section, it can be concluded that: i) compared with Naive and CRLUnsup, DaCoRL sacrifices a certain degree of sample efficiency, but achieves great performance improvement and ii) the performance of DaCoRL is also significantly better than that of CDKD and LLIRL, although the sample efficiency of these three methods is consistent.

VI. CONCLUSION AND FUTURE WORK

In this article, we present a CRL framework named DaCoRL as a viable solution for the agent to successfully conduct continual learning in dynamic environments. The goal of DaCoRL is to continuously adapt the RL agent's behavior toward the changing environment and to minimize the catastrophic forgetting of previously learned tasks. To this end, we employ an incremental context detection module to categorize the stream of tasks into a set of distinct contexts using online Bayesian infinite Gaussian mixture clustering. Then, a context-conditioned policy with an expandable multihead neural network is optimized, in conjunction with a knowledge distillation regularization term, to avoid interference among tasks both between and within contexts. A key profit of our method is that it can achieve incremental context instantiation in a fully online manner without requiring any external information to explicitly signal environmental changes. Meanwhile, it only relies on a single policy network to accomplish effective continual learning in dynamic environments and can be easily coupled with any other policy-based RL algorithms. Experiments on several continuous control tasks confirm that DaCoRL can significantly outperform state-of-the-art algorithms in terms of stability, overall performance, and generalization ability.

This article mainly addresses dynamic environment scenarios with abrupt changes between tasks. A promising direction for future work is to explore more challenging cases, including subtle variations from one task to another, or intensively changing environments where the changes may happen between consecutive episodes. Another direction is to develop an efficient strategy that can facilitate forward and backward transfer during the CRL training process.

REFERENCES

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

Authorized licensed use limited to: Bo Yuan. Downloaded on June 08,2023 at 13:42:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

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

[3] G. A. Rummery and M. Niranjan, *On-Line Q-Learning Using Connectionist Systems*. Cambridge, U.K.: Univ. of Cambridge, Department of Engineering Cambridge, 1994.

[4] W. Li, H. Luo, Z. Lin, C. Zhang, Z. Lu, and D. Ye, "A survey on transformers in reinforcement learning," 2023, *arXiv:2301.03044*.

[5] D. Ye et al., "Towards playing full MOBA games with deep reinforcement learning," in *Proc. Conf. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 621–632.

[6] S. Huang et al., "TiKick: Towards playing multi-agent football full games from single-agent demonstrations," 2021, *arXiv:2110.04507*.

[7] D. Ye et al., "Mastering complex control in MOBA games with deep reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 4, pp. 6672–6679.

[8] D. Ye et al., "Supervised learning achieves human-level performance in MOBA games: A case study of honor of kings," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 3, pp. 908–918, Mar. 2022.

[9] S. Chen, M. Zhu, D. Ye, W. Zhang, Q. Fu, and W. Yang, "Which heroes to pick? Learning to draft in MOBA games with neural networks and tree search," *IEEE Trans. Games*, vol. 13, no. 4, pp. 410–421, Dec. 2021.

[10] Z. Lin, J. Li, J. Shi, D. Ye, Q. Fu, and W. Yang, "JueWu-MC: Playing minecraft with sample-efficient hierarchical reinforcement learning," in *Proc. 31st Int. Joint Conf. Artif. Intell.*, Jul. 2022, pp. 3257–3263.

[11] A. Francis et al., "Long-range indoor navigation with PRM-RL," *IEEE Trans. Robot.*, vol. 36, no. 4, pp. 1115–1134, Aug. 2020.

[12] M. G. Bellemare et al., "Autonomous navigation of stratospheric balloons using reinforcement learning," *Nature*, vol. 588, no. 7836, pp. 77–82, Dec. 2020.

[13] M. A. K. 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.

[14] K. Khetarpal, M. Riemer, I. Rish, and D. Precup, "Towards continual reinforcement learning: A review and perspectives," 2020, *arXiv:2012.13490*.

[15] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," *Psychol. Learn. Motivat.*, vol. 24, pp. 109–165, Jan. 1989.

[16] R. French, "Catastrophic forgetting in connectionist networks," *Trends Cognit. Sci.*, vol. 3, no. 4, pp. 128–135, Apr. 1999.

[17] R. Hadsell, D. Rao, A. A. Rusu, and R. Pascanu, "Embracing change: Continual learning in deep neural networks," *Trends Cognit. Sci.*, vol. 24, no. 12, pp. 1028–1040, Dec. 2020.

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

[19] S. Kessler, J. Parker-Holder, P. Ball, S. Zohren, and S. J. Roberts, "UNCLEAR: A straightforward method for continual reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1–19.

[20] S. Padakandla and S. Bhatnagar, "Reinforcement learning algorithm for non-stationary environments," *Int. J. Speech Technol.*, vol. 50, no. 11, pp. 3590–3606, Nov. 2020.

[21] T. Zhang, X. Wang, B. Liang, and B. Yuan, "Catastrophic interference in reinforcement learning: A solution based on context division and knowledge distillation," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Apr. 19, 2022, doi: 10.1109/TNNLS.2022.3162241.

[22] C. Rasmussen, "The infinite Gaussian mixture model," in *Proc. Conf. Neural Inf. Process. Syst.*, vol. 12, 1999, pp. 1–7.

[23] J. Pitman, "Combinatorial stochastic processes," Dept. Stat., Univ. California, Berkeley, Berkeley, CA, USA, USA, Tech. Rep., 621, 2002.

[24] S. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "ICaRL: Incremental classifier and representation learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5533–5542.

[25] A. Rosenfeld and J. K. Tsotsos, "Incremental learning through deep adaptation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 3, pp. 651–663, Mar. 2020.

[26] M. Hersche, G. Karunaratne, G. Cherubini, L. Benini, A. Sebastian, and A. Rahimi, "Constrained few-shot class-incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 9047–9057.

[27] Z. Wang, H. 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.

[28] Z. Wang, C. Chen, and D. Dong, "Lifelong incremental reinforcement learning with online Bayesian inference," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 8, pp. 4003–4016, Aug. 2022.

[29] L. N. Alegre, A. L. Bazzan, and B. C. da Silva, "Minimum-delay adaptation in non-stationary reinforcement learning via online high-confidence change-point detection," in *Proc. Int. Conf. Auton. Agent. Multi. Agent Syst.*, 2021, pp. 97–105.

[30] L. Bottou, "Online learning and stochastic approximations," *Online Learn. Neural Netw.*, vol. 17, no. 9, p. 142, 1998.

[31] S. Shalev-Shwartz, "Online learning and online convex optimization," *Found. Trends Mach. Learn.*, vol. 4, no. 2, pp. 107–194, 2011.

[32] H. Huang, D. Ye, L. Shen, and W. Liu, "Curriculum-based asymmetric multi-task reinforcement learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 6, pp. 7258–7269, Jun. 2023.

[33] M. Crawshaw, "Multi-task learning with deep neural networks: A survey," 2020, *arXiv:2009.09796*.

[34] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 12, pp. 5586–5609, Dec. 2022.

[35] D. Isele and A. Cosgun, "Selective experience replay for lifelong learning," in *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1, pp. 1–8.

[36] M. Riemer et al., "Learning to learn without forgetting by maximizing transfer and minimizing interference," in *Proc. 7th Int. Conf. Learn. Represent.*, 2019, pp. 1–31.

[37] D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap, and G. Wayne, "Experience replay for continual learning," in *Proc. Conf. Neural Inf. Process. Syst.*, 2019, pp. 1–11.

[38] C. Atkinson, B. McCane, L. Szymanski, and A. Robins, "Pseudo-rehearsal: Achieving deep reinforcement learning without catastrophic forgetting," *Neurocomputing*, vol. 428, pp. 291–307, Mar. 2021.

[39] A. A. Rusu et al., "Policy distillation," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–13.

[40] R. Traore et al., "DisCoRL: Continual reinforcement learning via policy distillation," in *Proc. Conf. Neural Inf. Process. Syst. Workshop*, 2019, pp. 1–15.

[41] A. A. Rusu et al., "Progressive neural networks," 2016, *arXiv:1606.04671*.

[42] A. Mallya and S. Lazebnik, "PackNet: Adding multiple tasks to a single network by iterative pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7765–7773.

[43] Z. Wang, C. Chen, and D. Dong, "A Dirichlet process mixture of robust task models for scalable lifelong reinforcement learning," *IEEE Trans. Cybern.*, early access, May 17, 2022, doi: 10.1109/TCYB.2022.3170485.

[44] P. Auer, T. Jaksch, and R. Ortner, "Near-optimal regret bounds for reinforcement learning," in *Proc. Conf. Neural Inf. Process. Syst.*, vol. 21, 2008, pp. 1–8.

[45] P. Gajane, R. Ortner, and P. Auer, "A sliding-window algorithm for Markov decision processes with arbitrarily changing rewards and transitions," 2018, *arXiv:1805.10066*.

[46] R. Ortner, P. Gajane, and P. Auer, "Variational regret bounds for reinforcement learning," in *Proc. Conf. Uncertain. Artif. Intell.*, 2020, pp. 81–90.

[47] W. C. Cheung, D. Simchi-Levi, and R. Zhu, "Reinforcement learning for non-stationary Markov decision processes: The blessing of (more) optimism," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1843–1854.

[48] O. D. Domingues, P. Menard, M. Pirotta, E. Kaufmann, and M. Valko, "A kernel-based approach to non-stationary reinforcement learning in metric spaces," in *Proc. Int. Conf. Uncertain. Artif. Intell.*, 2021, pp. 3538–3546.

[49] B. C. da Silva, E. W. Basso, A. L. C. Bazzan, and P. M. Engel, "Dealing with non-stationary environments using context detection," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 217–224.

[50] E. Hadoux, A. Beynier, and P. Weng, "Sequential decision-making under non-stationary environments via sequential change-point detection," in *Proc. Int. Workshop Learn. Multiple Contexts*, 2014, pp. 1–11.

[51] G. Canonaco, M. Restelli, and M. Roveri, "Model-free non-stationarity detection and adaptation in reinforcement learning," in *Proc. Eur. Conf. Artif. Intell.*, 2020, pp. 1047–1054.

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

[53] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.

[54] H. Zhou, Z. Lin, J. Li, D. Ye, Q. Fu, and W. Yang, "Revisiting discrete soft actor-critic," 2022, *arXiv:2209.10081*.

[55] A. Nagabandi, C. Finn, and S. Levine, "Deep online learning via meta-learning: Continual adaptation for model-based RL," 2018, *arXiv:1812.07671*.

[56] V. Lomonaco, K. Desai, E. Culurciello, and D. Maltoni, "Continual reinforcement learning in 3D non-stationary environments," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 999–1008.

Authorized licensed use limited to: Bo Yuan. Downloaded on June 08,2023 at 13:42:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

[57] C. D'Eramo, D. Tateo, A. Bonarini, M. Restelli, and J. Peters, "Sharing knowledge in multi-task deep reinforcement learning," in *Proc. 8th Int. Conf. Learn. Represent.*, 2020, pp. 1–11.
[58] 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.
[59] Y. Seo, L. Chen, J. Shin, H. Lee, P. Abbeel, and K. Lee, "State entropy maximization with random encoders for efficient exploration," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 9443–9454.
[60] I. Goodfellow et al., "Generative adversarial nets," in *Proc. Conf. Neural Inf. Process. Syst.*, vol. 27, 2014.
[61] T. Yu et al., "Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Proc. Conf. Robot Learn.*, 2020, pp. 1094–1100.
[62] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[63] Z. Wu, C. Yu, D. Ye, J. Zhang, and H. H. Zhuo, "Coordinated proximal policy optimization," in *Proc. Conf. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 26437–26448.

Tiantian Zhang received the B.Sc. degree in automation from the Department of Information Science and Technology, Central South University, Changsha, China, in 2015, and the M.Sc. degree in control engineering from the Department of Automation, Tsinghua University, Beijing, China, in 2018, where she is currently pursuing the Ph.D. degree in control science and engineering.

Her research interests include data science, decision-making, and reinforcement learning.

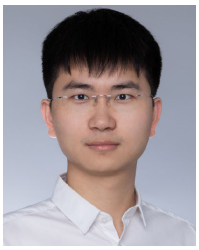
Zichuan Lin received the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2021.

He is now a Researcher at Tencent, Shenzhen, China, working on sample-efficient deep reinforcement learning algorithms. His interests include machine learning, reinforcement learning, as well as their applications on game AI and natural language processing.

Dr. Lin has been serving as a PC for NeurIPS, ICML, ICLR, and AAAI.

Yuxing Wang received the B.Sc. degree in communication engineering from the Department of Electronic Information, Southwest Minzu University, Chengdu, China, in 2020. He is currently pursuing the M.Sc. degree in electronic information with the Shenzhen International Graduate School, Tsinghua University, Shenzhen, China.

His research interests include evolutionary computation, reinforcement learning, and embodied AI.



Deheng Ye received the Ph.D. degree from the School of Computer Science and Engineering, Nanyang Technological University, Singapore, in 2016.

He is now a Principal Researcher and Team Manager with Tencent, Shenzhen, China, where he leads a group of engineers and researchers developing large-scale learning platforms and intelligent AI agents. He is interested in applied machine learning, reinforcement learning, and software engineering.

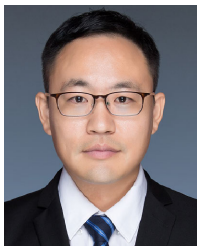
Dr. Ye has been serving as a PC/SPC for NeurIPS, ICML, ICLR, AAAI, and IJCAI.

Qiang Fu received the B.S. and M.S. degrees from the University of Science and Technology of China, Hefei, China, in 2006 and 2009, respectively.

He is the Director of the Game AI Center, Tencent AI Lab, Shenzhen, China. He has been dedicated to machine learning, data mining, and information retrieval for over a decade. His current research focus is game intelligence and its applications, leveraging deep learning, domain data analysis, reinforcement learning, and game theory.

Wei Yang received the M.S. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2007.

He is currently the General Manager of the Tencent AI Lab, Shenzhen, China. He has pioneered many influential projects at Tencent in a wide range of domains, covering Game AI, Medical AI, search, data mining, large-scale learning systems, and so on.



Xueqian Wang (Member, IEEE) received the M.Sc. and Ph.D. degrees in control science and engineering from the Harbin Institute of Technology (HIT), Harbin, China, in 2005 and 2010, respectively.

From June 2010 to February 2014, he was a Post-Doctoral Researcher with HIT. From March 2014 to November 2019, he was an Associate Professor with the Division of Informatics, Shenzhen International Graduate School, Tsinghua University, Shenzhen, China. He is currently a Professor and the Leader of the Center for Artificial Intelligence and Robotics, Shenzhen International Graduate School, Tsinghua University. His research interests include robot dynamics and control, teleoperation, intelligent decision-making and game-playing, and fault diagnosis.

Bin Liang (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in control engineering from the Honors College, Northwestern Polytechnical University, Xi'an, China, in 1989 and 1991, respectively, and the Ph.D. degree in control engineering from the Department of Precision Instrument, Tsinghua University, Beijing, China, in 1994.

From 1994 to 2003, he held positions of Post-Doctoral Researcher, Associate Researcher, and Researcher with the China Academy of Space Technology (CAST), Beijing. From 2003 to 2007, he held the positions of Researcher and Assistant Chief Engineer with the China Aerospace Science and Technology Corporation, Beijing. He is currently a Professor with the Research Center for Navigation and Control, Department of Automation, Tsinghua University. His research interests include modeling and control of intelligent robotic systems, teleoperation, and intelligent sensing technology.

Bo Yuan (Senior Member, IEEE) received the B.E. degree in computer science from the Nanjing University of Science and Technology, Nanjing, China, in 1998, and the M.Sc. and Ph.D. degrees in computer science from The University of Queensland (UQ), St Lucia, QLD, Australia, in 2002 and 2006, respectively.

From 2006 to 2007, he was a Research Officer on a project funded by the Australian Research Council, UQ. From 2007 to 2021, he was a Faculty Member (Lecturer, from 2007 to 2009, and Associate Professor, from 2009 to 2021) with the Division of Informatics, Tsinghua Shenzhen International Graduate School, Shenzhen, China, and served as the Deputy Director of the Office of Academic Affairs from 2013 to 2020. He has authored or coauthored more than 110 papers in refereed international conferences and journals. His research interests include data science, evolutionary computation, and reinforcement learning.

Xiu Li (Member, IEEE) received the Ph.D. degree in computer-integrated manufacturing from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2000.

From 2003 to 2010, she was an Associate Professor with the Department of Automation, Tsinghua University, Beijing, China. From 2010 to 2016, she was an Associate Professor with the Division of Informatics, Shenzhen International Graduate School, Tsinghua University, Shenzhen, China. She is currently a Professor and the Director of the Division of Informatics, Shenzhen International Graduate School, Tsinghua University. Her research interests include intelligent systems, pattern recognition, and data mining.

Authorized licensed use limited to: Bo Yuan. Downloaded on June 08,2023 at 13:42:06 UTC from IEEE Xplore. Restrictions apply.