Catastrophic Interference in Reinforcement Learning: A Solution Based on Context Division and Knowledge Distillation

Tiantian Zhang, Xueqian Wang, *Member, IEEE*, Bin Liang, *Senior Member, IEEE*, and Bo Yuan, *Senior Member, IEEE*

***Abstract*— The powerful learning ability of deep neural networks enables reinforcement learning (RL) agents to learn competent control policies directly from continuous environments. In theory, to achieve stable performance, neural networks assume identically and independently distributed (i.i.d.) inputs, which unfortunately does not hold in the general RL paradigm where the training data are temporally correlated and nonstationary. This issue may lead to the phenomenon of "catastrophic interference" and the collapse in performance. In this article, we present interference-aware deep Q-learning (IQ) to mitigate catastrophic interference in single-task deep RL. Specifically, we resort to online clustering to achieve on-the-fly context division, together with a multihead network and a knowledge distillation regularization term for preserving the policy of learned contexts. Built upon deep Q networks (DQNs), IQ consistently boosts the stability and performance when compared to existing methods, verified with extensive experiments on classic control and Atari tasks. The code is publicly available at https://github.com/Sweety-dm/Interference-aware-Deep-Q-learning.**

***Index Terms*— Catastrophic interference, context division, knowledge distillation, reinforcement learning (RL).**

I. Introduction

IN RECENT years, the successful application of deep neural networks (DNNs) in reinforcement learning (RL) [2] has provided a new perspective to boost its performance on high-dimensional continuous problems. With the powerful function approximation and representation learning capabilities of DNNs, deep RL is regarded as a milestone toward constructing autonomous systems with a higher level of understanding of the physical world [3]. Currently, deep RL has demonstrated great potential on complex tasks, from learning to play video games directly from pixels [4], [5] to making immediate decisions on robot behavior from camera inputs [6]–[8]. However, these successes are limited and prone to catastrophic interference[1] due to the inherent issue of DNNs in face of the nonstationary data distributions, and they rely heavily on a combination of various subtle strategies, such as experience replay [4] and fixed target networks [5], or distributed training architecture [9]–[11].

Catastrophic interference is the primary challenge for many neural network-based machine learning systems when learning over a nonstationary stream of data [12]. It is normally investigated in multitask continual learning (CL), mainly including supervised continual learning (SCL) for classification tasks [13]–[18] and continual reinforcement learning (CRL) [1], [13], [14], [19], [20] for decision tasks. In the multitask CL, the agent continually faces new tasks and the neural network may quickly fit to the data distribution of the current task while potentially overwriting the information related to learned tasks, leading to catastrophic forgetting of the solutions of old tasks. The underlying reason behind this phenomenon is the global generalization and overlapping representation of neural networks [21], [22]. Neural networks training normally assumes that the inputs are identically and independently distributed (i.i.d.) from a fixed data distribution and the output targets are sampled from a fixed conditional distribution. Only when this assumption is satisfied, can positive generalization be ensured among different batches of stochastic gradient descent. However, when the data distribution is drifted during training, the information learned from old tasks may be negatively interfered or even overwritten by the newly updated weights, resulting in catastrophic interference.

Deep RL is essentially a CL problem due to its learning mode of exploring while learning [20], and it is particularly vulnerable to catastrophic interference, even within many single-task settings where the environment is stationary (such as Atari 2600 games or even simpler classic control tasks in OpenAI Gym) [23]–[26]. The nonstationarity of data distributions in the single-task RL is mainly attributed to

Manuscript received August 4, 2021; revised December 11, 2021 and March 1, 2022; accepted March 16, 2022. This work was supported by the National Natural Science Foundation of China under Grant U1713214. *(Corresponding author: Bo Yuan.)*

Tiantian Zhang and Bo Yuan are with the Intelligent Computing Laboratory, Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China (e-mail: ztt19@mails.tsinghua.edu.cn; boyuan@ieee.org).

Xueqian Wang is with the Center for Artificial Intelligence and Robotics, Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China (e-mail: wang.xq@sz.tsinghua.edu.cn).

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

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TNNLS.2022.3162241.

Digital Object Identifier 10.1109/TNNLS.2022.3162241

[1] A phenomenon observed in neural networks where later training is likely to overwrite and interfere with previously learned good policies and significantly degrades the performance on previous tasks.

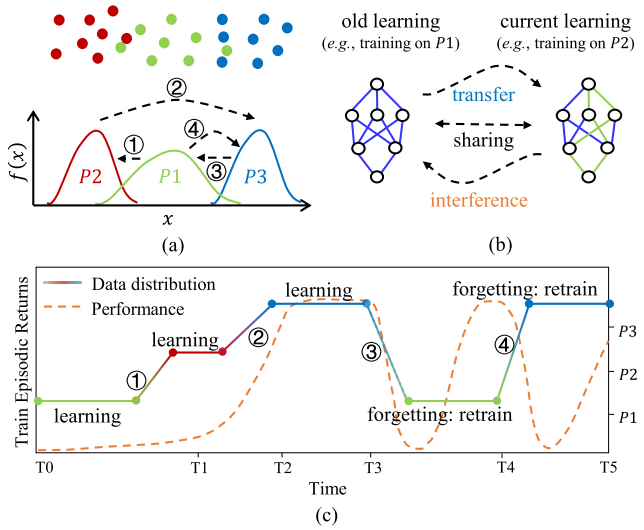


Fig. 1. Illustration of the catastrophic interference in the single-task RL. (a) Drift of data distributions during learning, where $P1$–$P3$ are different data distributions and ①–④ represent distribution transitions. The agent experiences the following data distribution transitions during learning: $P1 \xrightarrow{①} P2 \xrightarrow{②} P3 \xrightarrow{③} P1 \xrightarrow{④} P3$. (b) Stability–plasticity tradeoff in DNNs [1]. Sharing: both learning phases train the same model. Transfer: current learning phase continues training on the model derived from the old learning phase. Interference: after the model is trained on $P2$, the weights in green are changed in the right network, affecting the model performance on $P1$. (c) Learning curves where the solid line corresponds to the data distribution transitions in (a) and the dashed line shows the training performance. Before $T3$, the data distribution is gradually drifted from $P1$ to $P2$ and to $P3$. When the model fits to $P3$, the learned policies on $P1$ and $P2$ are interfered, resulting in catastrophically degraded performance when the agent encounters states from $P1$ again. Therefore, the model needs to be retrained on $P1$ (in the time period $T3 \rightarrow T4$). The same problem occurs in the time period $T4 \rightarrow T5$.

the following properties of RL. First, the inputs of RL are sequential observations received from the environment, which are temporally correlated. Second, in the progress of learning, the agent's decision-making policy changes gradually, which makes the observations nonstationary. Third, RL methods rely heavily on bootstrapping, where the RL agent uses its own estimated value function as the target, making the target outputs also nonstationary. In addition, as noted in [24], replay buffers with prioritized experience replay [27] that preferentially sample experiences with higher temporal-difference (TD) errors will also exasperate the nonstationarity of training data. Once the distribution of training data encounters notable drift, catastrophic interference and a chain reaction are likely to occur, resulting in a sudden deterioration of the training performance, as shown in Fig. 1.

Currently, there are two major strategies for dealing with catastrophic interference in the single-task RL training: experience replay [4], [5] and local optimization [25], [26]. The former usually exhibits high-level sensitivity to key parameters (e.g., replay buffer capacity) and often requires maintaining a large experience storage memory. Furthermore, the sufficiently large memory may increase the degree of off-policyness of transitions in the buffer [28], violating the requirement of current state-of-the-art algorithms that the data should be close to the on-policy distribution, even for off-policy algorithms such as deep Q network (DQN). The latter advocates local network updating for the data with a specific distribution instead of global generalization to reduce the representation overlap among different data distributions. The major issues are that some methods are limited in the capability of model transfer among differently distributed data [25], [26] or require pretraining and may not be suitable for the online settings [26].

In this article, we focus on the catastrophic interference problem caused by state distribution drift in the single-task RL. We propose an interference-aware scheme with low buffer-size sensitivity called interference-aware deep Q-learning (IQ)[2] that estimates the value function online for each state distribution by minimizing the weighted sum of the original loss function of RL algorithms and the regularization term regarding the interference among different groups of states. The schematic architecture is shown in Fig. 3.

In order to mitigate the interference among different state distributions during model training, we introduce the concept of "context" into the single-task RL and propose a context division strategy based on online clustering. We show that it is essential to decouple the correlations among different state distributions with this strategy to divide the state space into a series of independent contexts (each context is a set of states distributed close to each other, conceptually similar to "task" in the multitask CRL). To achieve efficient and adaptive partition, we employ sequential K-means clustering [29] to process the states encountered during training in real time. Then, we parameterize the value function by a neural network with multiple output heads commonly used in multitask learning [19], [30], [31] in which each output head specializes on a specific context, and the feature extractor is shared across all contexts. In addition, we apply knowledge distillation as a regularization term in the objective function for value function estimation, which can preserve the learned policies, while the RL agent is trained on states guided by the current policy, to further avoid the interference caused by the shared low-level representation. Furthermore, to ease the curse of dimensionality in high-dimensional state spaces, we employ a random encoder as its low-dimensional representation space can effectively capture the information about the similarity among states without any representation learning [32]. Clustering is then performed in the low-dimensional representation space of the randomly initialized convolutional encoder.

The contributions of this article are summarized as follows.

1) A novel context division strategy is proposed for the single-task RL. It is essential as the widely studied multitask CRL methods cannot be used directly to reduce interference in the single-task RL due to the lack of predefined task boundaries. This strategy can detect contexts adaptively online so that each context can be regarded as a task in multitask settings. In this way, the strategies designed for the multitask CRL can be used in the single-task RL to mitigate catastrophic interference.
2) A novel RL training scheme called IQ based on multi-head neural networks is proposed following the context

[2]So named because it uses deep Q-learning and features interference awareness. IQ also implies a smarter agent in the sense that it is the abbreviation of intelligence quotient.

division strategy. By incorporating the knowledge distillation loss into the objective function, IQ can better alleviate the interference suffered in the single-task RL than existing methods in a fully online manner.

3) A fixed random encoder is introduced into the context division of high-dimensional state spaces, which further stabilizes the performance of IQ on complex RL tasks (e.g., image-level inputs) compared with the underlying RL trained encoder.
4) Extensive experiments on a suite of OpenAI Gym standard benchmark environments ranging from classic control tasks to high-dimensional complex arcade learning environments (ALEs) [33] under various replay buffer capacity settings are conducted to validate that the overall superiority of our method over baselines in terms of the stability and the maximum achieved cumulative reward.

The rest of this article is organized as follows. Section II reviews the relevant strategies for alleviating catastrophic interference as well as context detection and identification. Section III introduces the nature of RL in terms of CL and gives an example analysis of catastrophic interference in the single-task RL. The details of IQ are shown in Section IV, and experimental results and analyses are presented in Section V. Finally, this article is concluded in Section VI with some discussions and directions for future work.

II. Related Work

Catastrophic interference within the single-task RL is a special case of CRL, which involves not only the strategies to mitigate interference but also the context detection and identification techniques.

A. Multitask Continual Reinforcement Learning

Multitask CRL has been an active research area with the development of RL architectures [34]. Existing methods mainly consist of three categories: experience replay-, regularization-, and parameter isolation-based methods.

The core idea of experience replay is to store samples of previous tasks in raw format (e.g., selective experience replay (SER) [35], meta experience replay (MER) [1], and continual learning with experience and replay (CLEAR) [36]) or generate pseudo-samples from a generative model (e.g., reinforcement pseudo rehearsal (RePR) [37]) to maintain the knowledge about the past in the model. These previous task samples are replayed while learning a new task to alleviate interference, in the form of either being reused as model inputs for rehearsal [35], [37] or constraining the optimization of the new task loss [1], [36]. Experience replay has become a very successful approach to tackling interference in CRL. However, the raw format may result in significant storage requirements for complex CRL settings. Although the generative model can be exempted from a replay buffer, it is still difficult to capture the overall distribution of previous tasks.

Regularization-based methods avoid storing raw inputs by introducing an extra regularization term into the loss function to consolidate previous knowledge while learning on new tasks. The regularization term includes penalty computing and knowledge distillation. The former focuses on reducing the chance of weights being modified. For example, elastic weight consolidation (EWC) [13] and UNcertainty guided Continual LEARning (UNCLEAR) [19] use the Fisher matrix to measure the importance of weights and protect important weights on new tasks. The latter is a form of knowledge transfer [38], which expects that the model trained on a new task can still perform well on the old ones. It is often used for policy transfer from one model to another (e.g., policy distillation [39], genetic policy optimization (GPO) [40], and distillation for continual reinforcement learning (DisCoRL) [41]). This family of solutions is easy to implement and tends to perform well on a small number of tasks but still faces challenges as the number of tasks increases.

Parameter isolation-based methods dedicate different model parameters to each task, to prevent any possible interference among tasks. Without the constraints on the size of neural networks, one can grow new branches for new tasks while freezing previous task parameters (e.g., progressive natural networks (PNNs) [42]). Alternatively, the architecture remains static, with fixed parts being allocated to each task. For instance, PathNet [14] uses a genetic algorithm to find a path from input to output for each task in the neural network and isolates the used network parts in parameter level from the new task training. These methods typically require networks with enormous capacity, especially when the number of tasks is large, and there is often unnecessary redundancy in the network structure, bringing a great challenge to model storage and efficiency.

B. Single-Task RL

Compared with the multitask CRL, catastrophic interference in the single-task RL remains an emerging research area, which has been relatively underexplored. There are two primary aspects of previous studies: one is finding supporting evidence to confirm that catastrophic interference is indeed prevalent within a specific RL task and the other is proposing effective strategies for dealing with it.

Researchers in DeepMind studied the learning dynamics of the single-task RL and developed a hypothesis that the characteristic coupling between learning and data generation is the main cause of interference and performance plateaus in deep RL systems [23]. Recent studies further confirmed this hypothesis and its universality in the single-task RL through large-scale empirical studies (called memento experiments) in Atari 2600 games [24]. However, none of these studies has suggested any practical solution for tackling the interference.

In order to mitigate interference, many deep RL algorithms, such as DQN [5] and its variants (e.g., double DQN [43] and Rainbow [44]), employ experience replay and fixed target networks to produce approximately i.i.d. training data, which may quickly become intractable in terms of memory requirement as task complexity increases. Furthermore, even with sufficient memory, it is still possible to suffer from catastrophic interference due to the imbalanced distribution of experiences.

In recent studies [21], [25], [26], researchers proposed some methods based on local representation and optimization

of neural networks, which showed that interference can be reduced by promoting the local updating of weights while avoiding global generalization. Sparse representation neural network (SRNN) [26] induces sparse representations in neural networks by introducing a distributional regularizer, which requires a large batch of data generated by a fixed policy that covers the space for pretraining and has not been extended to the online setting. Dynamic self-organizing map (DSOM) [25] with neural networks introduces a DSOM module to induce such locality updates. These methods can reduce interference to some extent, but they may inevitably suffer from the lack of positive transfer in the representation layer and require larger network capacity, which is not desirable in complex tasks. Recently, discretizing neural network (D-NN) and tile coding neural network (TC-NN) were used to remap the input observations to a high-dimensional space to sparsify input features, reducing the activation overlap [21]. However, tile coding increases the dimension of inputs to a neural network, which can lead to scalability issues for spaces with high dimensionality.

C. Context Detection and Identification

It is a fundamental step for learning task relatedness in CL. Most multitask CL methods aforementioned rely on well-defined task boundaries and are usually trained on a sequence of tasks with known labels or boundaries. Existing context detection approaches commonly leverage statistics or Bayesian inference to identify task boundaries.

On the one hand, some methods tend to be reactive to a changing distribution by finding change points in the pattern of state–reward tuples (e.g., Context QL [45]), tracking the difference between the short-term and long-term moving average rewards (e.g., CRL-Unsup [46]), or splitting a game into contexts using the undiscounted accumulated game score as a task contextualization [47]. These methods can be agile in responding to scenarios with abrupt changes among contexts or tasks but are insensitive to smooth transitions from one context to another.

On the other hand, some more ambitious approaches try to learn a belief of the unobserved context state directly from the history of environment interactions, such as forget-me-not process (FMN) [48] for piecewise-repeating data generating sources and continual unsupervised representation learning (CURL) [49] for task inference without any knowledge about task identity. However, they both need to be pretrained with the complete data before applied to CL problems, and CURL itself also needs additional techniques to deal with the interference.

Furthermore, Ghosh *et al.* [50] proposed to partition the initial state space into a finite set of contexts by performing a K-means clustering procedure, which can decompose more complex tasks, but cannot completely decouple the correlations among different state distributions from the perspective of interference prevention.

III. Preliminaries and Problem Statement

To better characterize the problem studied in this article, some key definitions and glossaries of CRL problems are introduced in this section.

A. Definitions and Glossaries

Some important definitions of RL relevant to this article are presented as follows.

Definition 1 (RL Paradigm [2]): An RL problem is regarded as a Markov decision process (MDP), 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 environment 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.

According to Definition 1, 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 taking action A_t and receives reward $R(S_t, A_t)$. Based on this definition, the optimization objective of value-based RL models is defined as follows.

Definition 2 (RL Optimization Objective [20]): The optimization objective of the value-based RL is to learn a policy $\pi(a|s)$ with internal parameter $\theta \in \Theta$ that maximizes the expected long-term discounted returns for each (s, a) in time, also known as the value function

$$J(\pi) = Q_\pi(s, a) = \mathbb{E}_{\mathcal{P},\pi}\left[\sum_{k=0}^{\infty} \gamma^k R(S_{t+k}, A_{t+k}) \middle| S_t = s, A_t = a\right]. \quad (1)$$

Here, the expectation is over the process that generates a history using $\mathcal{P}$ and decides actions from π until the end of the agent's lifetime.

The optimization objective in Definition 2 does not just concern itself with the current state but also the full expected future distribution of states. As such, it is possible to overcome the catastrophic interference for RL over nonstationary data distributions. However, much of the recent work in RL has been in the so-called episodic environments, which optimizes the episodic RL objective:

Definition 3 (Episodic RL Optimization Objective [20]): Given some future horizon H, find a policy $\pi(a|s)$, optimizing the expected discounted returns

$$J_{\text{episodic}}(\pi) = Q_\pi(s, a) = \mathbb{E}_{\mathcal{P},\pi}\left[\sum_{k=0}^{H-1} \gamma^k R(S_{t+k}, A_{t+k}) \middle| S_t = s, A_t = a\right]. \quad (2)$$

Here, to ensure the feasibility and ease of implementation of optimization, the objective is only optimized over a future horizon H until the current episode terminates.

It is clear that the episodic objective in Definition 3 is biased toward the current episode distribution while ignoring the possibly far more important future episode distributions over the agent's lifetime. Plugging in such an objective directly into the nonstationary RL settings leads to biased optimization, which is likely to cause catastrophic interference effects.

For large-scale domains, the value function is often approximated with a member of the parametric function class, such as a neural network with parameter $\theta \in \Theta$, expressed as $Q(s, a; \theta)$, which is fit online using experience samples of the form (s, a, r, s'). This experience is typically collected into

a buffer $\mathcal{B}$ from which batches are later drawn at random to form a stochastic estimate of the loss

$$\mathcal{L}(\theta) = \mathbb{E}_{\mu}\left[L\left(r + \gamma \max_{a' \in \mathcal{A}} Q(s', a'; \theta^{-}) - Q(s, a; \theta)\right)\right] \quad (3)$$

where $L : \mathbb{R} \to \mathbb{R}$ is the agent's loss function and $\mu \in \mathcal{P}(\mathcal{B})$ is the distribution that defines its sampling strategy. In general, the parameter θ^{-} used to compute the target $Q(s', a'; \theta^{-})$ is a prior copy of that used for action selection (as the settings of DQN [5]).

In addition, it is necessary to clarify some important glossaries in relation to CL.

1) Nonstationary [51]: It is a process whose state or probability distribution changes with time.

2) Interference [22]: It is a type of influence between two gradient-based processes with objectives J_1 and J_2 and sharing parameter θ. Interference is often characterized in the first order by the inner product of their gradients

$$\rho_{1,2} = \nabla_{\theta} J_1^{T} \nabla_{\theta} J_2 \quad (4)$$

and can be seen as being constructive ($\rho > 0$, transfer) or destructive ($\rho < 0$, interference), when applying a gradient update using $\nabla_{\theta} J_1$ on the value of J_2.

3) Catastrophic Interference [12], [51]: A phenomenon observed in neural networks training where learning a new task significantly degrades the performance on previous tasks.

B. Problem Statement

The interference within the single-task RL can be approximately measured by the difference in TD errors before and after model update under the current policy, referred to as approximate expected interference (AEI) [52]

$$\text{AEI} = \mathbb{E}_{\hat{d}}\left[\delta(s, a, r, s'; \theta_t)^2 - \delta(s, a, r, s'; \theta_{t-1})^2\right] \quad (5)$$

where $\hat{d}$ is the distribution of (s, a, r, s') under the current policy and $\delta(s, a, r, s'; \theta) = r + \gamma \max_{a' \in \mathcal{A}} Q(s', a'; \theta) - Q(s, a; \theta)$ is the TD error.

To illustrate the interaction between interference and the agent's performance during the single-task RL training, we run an experiment on CartPole using the DQN implemented in OpenAI Baselines[3] and set the replay buffer size N to 100, a small capacity to trigger interference to highlight its effect. We trained the agent for 300k environment steps and approximated $\hat{d}$ with a buffer containing recent transitions of capacity 10k to evaluate the AEI value according to (5) after each update. Fig. 2 shows two segments of the interference and performance curves during training from which we can see that the performance started to oscillate when AEI started to increase [e.g., $t \approx 118$k, $t \approx 143$k, and $t \approx 175$k in Fig. 2(a) and $t \approx 230$k in Fig. 2(b)]. In general, the performance of the agent tends to drop significantly in the presence of increasing interference. This result provides direct evidence that interference is correlated closely with the stability of the single-task RL model.

[3]OpenAI Baselines is a set of high-quality implementations of RL algorithms implemented by OpenAI: https://github.com/openai/baselines

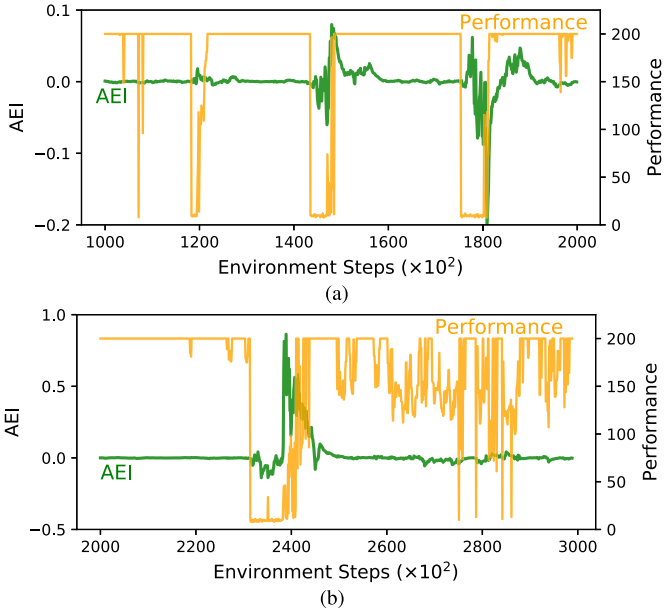


Fig. 2. Interference (green) and training performance (yellow) curve segments of a DQN agent on CartPole ($N = 100$). The interference is measured as the expectation in (5) and the performance is evaluated by the sum of discounted reward per episode. (a) Phase I with $t = 100$k–200k. (b) Phase II with $t = 200$k–300k.

From the analysis above, we state the problem investigated in this article as: proposing a novel and effective training scheme for the single-task RL, to reduce catastrophic interference and performance oscillation during training, improving the stability and overall performance simultaneously.

IV. Proposed Method

In this section, we give a detailed description of our IQ scheme whose architecture is shown in Fig. 3. IQ consists of three main components, which are jointly optimized to mitigate catastrophic interference in the single-task RL: context division, knowledge distillation, and the collaborative training of the multihead neural network. Based on IQ, we further propose interference-aware deep Q-learning with random encoder (IQ-RE) for the efficient contextualization of high-dimensional state spaces.

As mentioned before, catastrophic interference is an undesirable byproduct of global updates to the neural network weights on data whose distribution changes over time. A rational solution to this issue is to estimate an individual value function for each distribution, instead of using a single value function for all distributions. When an agent updates its value estimation of a state, the update should only affect the states within the same distribution. With this intuition in mind, we adopt a multihead neural network with shared representation layers to parameterize the distribution-specific value functions.

The IQ scheme proposed in this article can be incorporated into any existing value-based RL methods to train a piecewise Q-function for the single-task RL. The neural network is parameterized by a shared feature extractor and a set of linear output heads, corresponding to each context. As shown in Fig. 3, the set of weights of the Q-function is denoted by

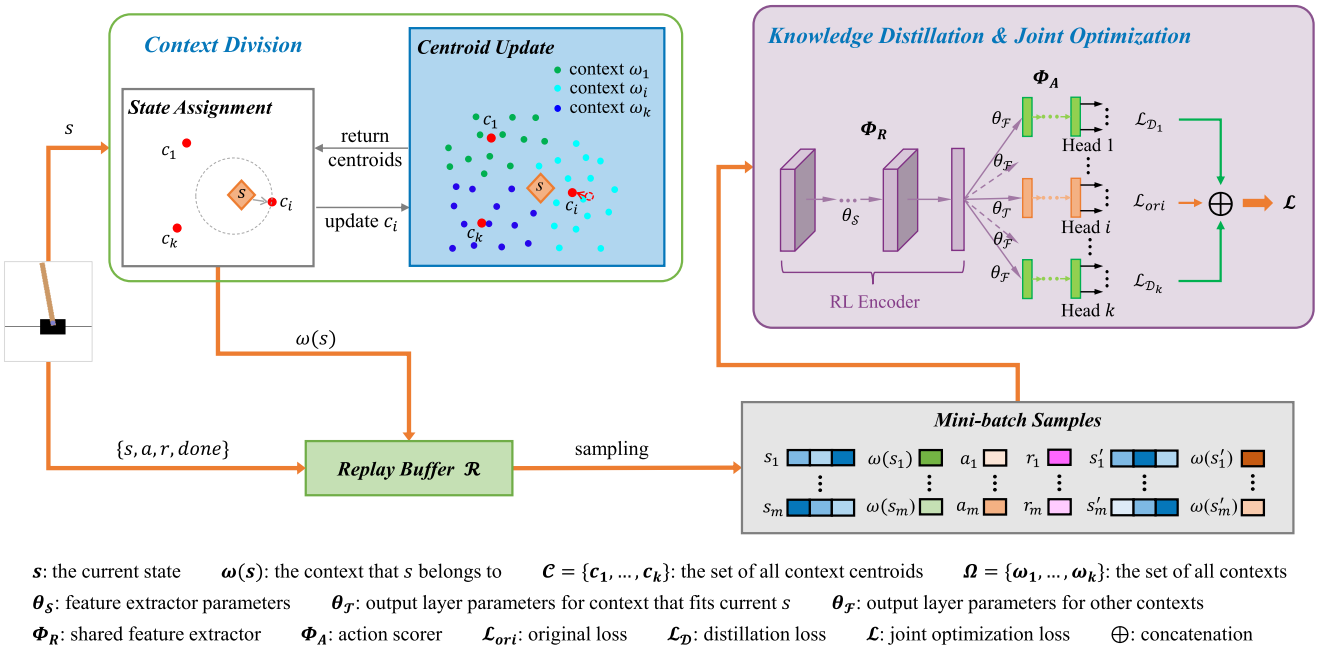


Fig. 3. Overview of the IQ scheme. This framework consists of three components: 1) context division, including state assignment and centroid update, and adaptive context division is achieved using sequential K-means clustering online; 2) knowledge distillation—the knowledge distillation loss ($\mathcal{L}_{\mathcal{D}}$) is incorporated into the objective function ($\mathcal{L}_{\text{ori}}$) to avoid interference among contexts due to the shared feature extractor; and 3) joint optimization with a multihead neural network, which aims to estimate the value for each $[s, a, \omega(s)]$ with the joint optimization loss ($\mathcal{L} = \mathcal{L}_{\text{ori}} + \lambda\mathcal{L}_{\mathcal{D}}$). Here, to keep consistency with the random encoder introduced later, we also call the representation module of the neural network as "RL encoder." In summary, our method can improve the performance by decoupling the correlations among differently distributed states and intentionally preserving the learned policies.

$\theta = \{\theta_{\mathcal{S}}, \theta_{\mathcal{T}}, \theta_{\mathcal{F}}\}$, where $\theta_{\mathcal{S}}$ is a set of shared parameters, while $\theta_{\mathcal{T}}$ and $\theta_{\mathcal{F}}$ are both context-specific parameters: $\theta_{\mathcal{T}}$ is for the context that corresponds to the current input state s and $\theta_{\mathcal{F}}$ is for others. In this section, we take the combination of IQ and the basic RL algorithm DQN as an illustrative example.

A. Context Division

In MDPs, states (or "observations") represent the most comprehensive information regarding the environment. To better understand the states of different distributions, we define a variable ω for a set of states that are close to each other in the state space, referred to as "context." Formally,

$$\begin{aligned} \Omega &= (\omega_i)_{i=1}^{k} \\ \mathcal{S} &= \cup_{i=1}^{k} \mathcal{S}_i \end{aligned} \tag{6}$$

where Ω is a finite set of contexts and k is the number of contexts. For an arbitrary MDP, we partition its state space into k contexts, and all states within each context follow approximately the same distribution, to decouple the correlations among states against distribution drift. More precisely, for a partition of $\mathcal{S}$ in (6), we associate a context ω_i with each set $\mathcal{S}_i$, so that for $s \in \mathcal{S}_i$, $\omega(s) = \omega_i$, where $\omega(s)$ can be thought of as a function of state s.

The inherent learning-while-exploring feature of RL agents leads to the fact that the agent generally does not experience all possible states of the environment while searching for the optimal policy. Thus, it is unnecessary to process the entire state space. Based on this fact, in IQ, we only perform context division on states experienced during training. In this article, we employ sequential K-means clustering [29] (see Appendix A-A of the supplementary material) to achieve context detection adaptively.

In Fig. 3, k centroids $\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$ are initialized at random in the entire state space. In each subsequent time step t, we execute state assignment and centroid update steps for each incoming state received from the environment[4] and store its corresponding transition $\{s_t, \omega(s_t), a_t, r_t, s_{t+1}, \omega(s_{t+1})\}$ into the replay buffer $\mathcal{B}$. Accordingly, in the training phase, we randomly sample a batch of transitions from $\mathcal{B}$ and train the shared feature extractor Φ_R and the specific output head Φ_A corresponding to the input state simultaneously while conducting fine-tuning of other output heads to avoid interference on learned policies. Since we store the context label of each state in the replay buffer, there are no additional state assignments required at every update step.[5]

Note that it is also possible to conduct context division based on the initial state distribution [50]. By contrast, we show that the partition of all states experienced during training can produce more accurate and effective context division results, as the trajectories starting from the initial states within different contexts have a high likelihood of overlapping in the subsequent time steps (see Appendix A-B of the supplementary material for more details).

1) Interference Among Contexts: We investigate the interference among contexts obtained by our context division method in detail. Specifically, we measure the Huber loss of TD errors in different contexts of the game as the agent

[4]Note that it is suggested to normalize the state in different dimensions before performing these two steps for more reasonable context division results.

[5]In IQ, we only need to perform state assignment once for each state.

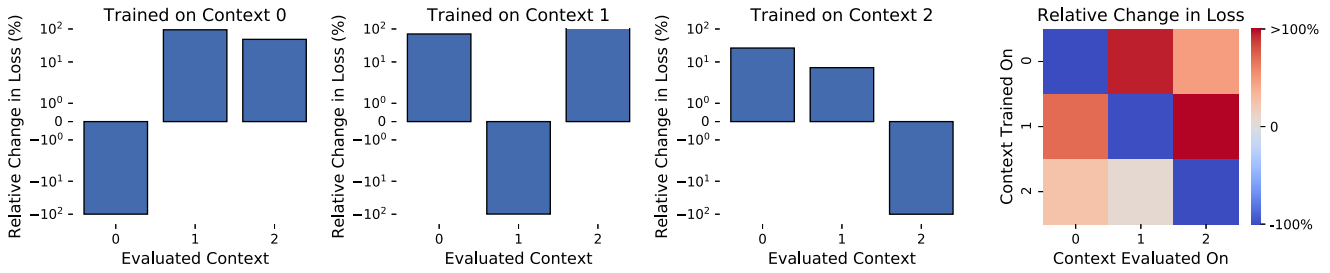


Fig. 4. Measuring the interference among contexts by clustering all experienced states when the agent is trained on CartPole-v0 for 400k environment steps ($k = 3$). We record the relative changes in the Huber loss for all contexts when the agent is trained on a particular context. It is clear that training on a particular context generally reduces the loss on itself and increases the losses on all other contexts.

learns in other contexts and then record the relative changes in loss before and after the agent's learning, as shown in Fig. 4. The results show that long-term training on any context may lead to negative generalization on all other contexts, even in such simple RL task CartPole-v0. The results on Pendulum-v0 shown in Appendix A-C of the supplementary material also support the same conclusion.

2) Computational Complexity: Assuming a d-dimensional environment of k contexts, the time and space complexities of our proposed context division module to process T environment steps are $\mathcal{O}(Tkd)$ and $\mathcal{O}(kd)$, respectively.

B. Knowledge Distillation

The shared low-level representation can cause the learning in new contexts to interfere with previous learning results, leading to catastrophic interference. A relevant technique to address this issue is knowledge distillation [38], which works well for encouraging the outputs of one network to approximate the outputs of another. The concept of distillation was originally used to transfer knowledge from a complex ensemble of networks to a relatively simpler network to reduce model complexity and facilitate deployment. In IQ, we use it as a regularization term in the value function estimation to preserve the previously learned information.

When training the model on a specific context, we need to consider two aspects of the loss function: the general loss of the current training context (denoted by $\mathcal{L}_{\text{ori}}$) and the distillation loss of other contexts (denoted by $\mathcal{L}_{\mathcal{D}}$). The former encourages the model to adapt to the current context to ensure plasticity, while the latter encourages the model to keep the memory of other contexts, preventing interference.

To incorporate IQ into the DQN framework, we rewrite the original loss function of DQN in (3) with the context variable ω as

$$\mathcal{L}_{\text{ori}}(\theta_{\mathcal{S}}, \theta_{\mathcal{T}}) = \mathbb{E}_{\mu}\Big[L\Big(Q_{\tau} - Q(s, a, \omega(s); \theta_{\mathcal{S}}, \theta_{\mathcal{T}})\Big)\Big] \tag{7}$$

where

$$Q_{\tau} = r + \gamma \max_{a' \in \mathcal{A}} Q\big(s', a', \omega(s'); \theta_{\mathcal{S}}^{-}, \theta_{\mathcal{T}}^{-}\big) \tag{8}$$

is the estimated target value of $Q(s, a, \omega(s); \theta_{\mathcal{S}}, \theta_{\mathcal{T}})$, μ is the distribution of samples, i.e., $\{s, \omega(s), a, r, s', \omega(s')\} \sim \mu$, and L refers to the Huber loss.

Algorithm 1 IQ

Input: Initial replay buffer $\mathcal{B}$ with capacity $|\mathcal{B}| = N$;
Initial Q-function f_θ with random weights θ;
Initial target $\hat{Q}$-function f_{θ^-} with weights $\theta^- = \theta$;
Initial context centroids $\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$;
Initial target context centroids $\hat{\mathcal{C}} = \mathcal{C}$.
Parameter: Total training steps T, the number of contexts k, target update period C, learning rate α.
Output: Updated $\mathcal{C}$ and f_θ.

Initial state s;
for $t = 1, T$ **do**
  Interact with environment to obtain $\{s_t, a_t, r_t, s_{t+1}\}$.
  States assignment: $\omega(s_t) \overset{\hat{\mathcal{C}}}{\leftarrow} s_t$, $\omega(s_{t+1}) \overset{\hat{\mathcal{C}}}{\leftarrow} s_{t+1}$.
  Store transition $\{s_t, \omega(s_t), a_t, r_t, s_{t+1}, \omega(s_{t+1})\}$ in $\mathcal{B}$.
  Context centroids update: $\mathcal{C} \leftarrow SKM(s_t, \mathcal{C})$.
  Joint optimization:
    Sample mini-batch $\{s_i, \omega(s_i), a_i, r_i, s_i', \omega(s_i')\}_{i=1}^{m}$;
    Calculate $\mathcal{L}_{ori}$, $\mathcal{L}_{\mathcal{D}}$ according to Eqs. (7) and (9);
    Perform a gradient descent step on Eq. (11) *w.r.t.* θ:
    $\theta \leftarrow \theta - \alpha \nabla_\theta (\mathcal{L}_{ori} + \lambda \mathcal{L}_{\mathcal{D}})$.
  Reset $\theta^- = \theta$ and $\hat{\mathcal{C}} = \mathcal{C}$ every C training steps.
end for

For each of the other contexts that the environment contains, we expect the output value for each pair of (s, a) to be close to the recorded output from the original network. In knowledge distillation, we regard the learned Q-function before the current update step as the teacher network, expressed as $Q_t^i = Q(s, a, \omega_i; \theta_{\mathcal{S}}^{-}, \theta_{\mathcal{F}}^{-})$, and the current network to be trained as the student network, expressed as $Q_s^i = Q(s, a, \omega_i; \theta_{\mathcal{S}}, \theta_{\mathcal{F}})$, where $\omega_i \in \Omega$ except the current context $\omega(s)$. Thus, the distillation loss is defined as

$$\mathcal{L}_{\mathcal{D}}(\theta_{\mathcal{S}}, \theta_{\mathcal{F}}) = \mathbb{E}_{\mu} \sum_{\omega_i \neq \omega(s), \omega_i \in \Omega} \mathcal{L}_{\omega_i}(\theta_{\mathcal{S}}, \theta_{\mathcal{F}}) \tag{9}$$

where

$$\mathcal{L}_{\omega_i}(\theta_{\mathcal{S}}, \theta_{\mathcal{F}}) = L\big(Q_t^i - Q_s^i\big) \tag{10}$$

is the distillation loss function of the output head corresponding to context ω_i.

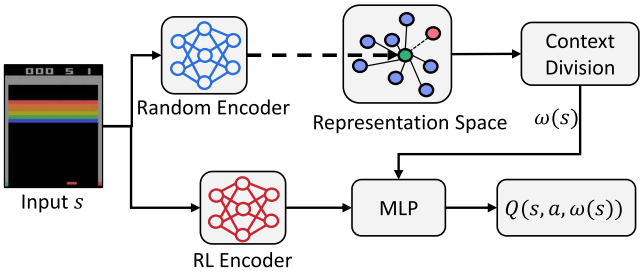


Fig. 5. Illustration of IQ-RE. The context division is performed in the low-dimensional representation space of a random encoder. A separate RL encoder is used to work with the MLP layers to estimate the value function.

C. Joint Optimization Procedure

To optimize a Q-function that can guide the agent to make proper decisions on each context without being adversely affected by catastrophic interference, we combine (7) and (9) to form a joint optimization framework. Namely, we solve the catastrophic interference problem by the following optimization objective:

$$\min_{\theta_{\mathcal{S}},\theta_{\mathcal{T}},\theta_{\mathcal{F}}} \mathcal{L}_{\text{ori}}(\theta_{\mathcal{S}},\theta_{\mathcal{T}}) + \lambda \mathcal{L}_{\mathcal{D}}(\theta_{\mathcal{S}},\theta_{\mathcal{F}}) \tag{11}$$

where $\lambda \in [0, 1]$ is a coefficient to control the tradeoff between the stability and plasticity of the neural network.

The complete procedure is described in Algorithm 1. The proposed method performs the context division in parallel to the training process without requiring additional data. For network training, to reduce the correlations with the target and ensure the stability of model training, the target network parameter θ^- is only updated by the Q-network parameter θ every C steps and is held fixed between individual updates, as in DQN [5]. Similarly, we also adopt fixed target context centroids ($\hat{\mathcal{C}}$) to avoid a small amount of instability of states assignment step introduced by constantly updated context centroids ($\mathcal{C}$). To simplify the model implementation, we set the updating frequency of the target context centroids to be consistent with the target network.

D. Random Encoders for High-Dimensional State Space

For high-dimensional state spaces, we propose to use random encoders for efficient context division, which can map high-dimensional inputs into low-dimensional representation spaces, overcoming the "curse of dimensionality." Although the original RL model already contains an encoder module, it is constantly updated and directly performing clustering in its representation space may introduce extra instability into context division. Therefore, based on IQ, we exploit a dedicated random encoder module for dimension reduction. Fig. 5 gives an illustration of this updated framework called IQ-RE in which the structure of the random encoder $f_{\theta_{re}}$ is consistent with the underlying RL encoder, but its parameter θ_{re} is randomly initialized and fixed throughout training. We provide the full procedure of IQ-RE in Appendix B-B of the supplementary material.

The main motivation of using random encoders arises from the observation that distances in the representation space of random encoder are adequate for finding similar states without any representation learning [32], that is, the representation space of a random encoder can effectively capture the information about the similarity among states without any representation learning (see Appendix B-A of the supplementary material). Additional comparative experiments of IQ with the random encoder and the underlying RL trained encoder in Appendix B-B of the supplementary material further highlight the superiority of random encoders.

V. Experiments and Evaluations

In this section, we conduct comprehensive experiments on several standard benchmarks from OpenAI Gym[6] containing four classic control tasks and six high-dimensional complex Atari games to demonstrate the effectiveness of our method.

A. Datasets

Classic control [53] contains four classic control tasks: CartPole-v0, Pendulum-v0, CartPole-v1, and Acrobot-v1, where the dimensions of state spaces are in the range of 3–6. The maximum time steps are 200 for CartPole-v0 and Pendulum-v0 and 500 for CartPole-v1 and Acrobot-v1. Meanwhile, the reward thresholds used to determine tasks solved are 195.0, 475.0, and -100.0 for CartPole-v0, CartPole-v1, and Acrobot-v1, respectively, while that for Pendulum-v0 is not yet specified. We choose these commonly used domains as they are well-understood and relatively simple, suitable for highlighting the mechanism and verifying the effectiveness of our method in a straightforward manner.

Atari games [33] contain six image-level complex tasks: Pong, Breakout, Carnival, Freeway, Tennis, and FishingDerby, where the observation is the screenshot represented by an RGB image of size $210 \times 160 \times 3$. We choose these domains to further demonstrate the scalability of our method on high-dimensional complex tasks that present significant challenges for existing baseline methods.

B. Implementation

1) Network Structure: For the four classic control tasks, we employ a fully connected layer as the feature extractor and a fully connected layer as the multihead action scorer, following the network configuration for this type of tasks in OpenAI Baseline. For the six Atari games, we employ the similar convolution neural network as in [44] and [54] for feature extracting and two fully connected layers as the multihead action scorer. More details can be found in Appendix C of the supplementary material.

2) Parameter Setting: In IQ, there are two key parameters: λ and k. To simplify parameter setting, we set λ in accordance with the exploration proportion ϵ in all experiments: $\lambda = 1-\epsilon$, due to the inverse relationship between them in training. In the early training, ϵ is close to 1, and the model is normally inaccurate with little interference, and a small λ (close to 0) can promote plasticity construction of the model. Then,

[6]OpenAI Gym is a publicly available released implementation repository of RL environments: https://github.com/openai/gym

ϵ gradually approaches 0 during the subsequent training, and the model has learned more useful information, while interference is also likely to occur. Consequently, smoothly increasing λ is needed to ensure plasticity while avoiding interference. Meanwhile, we set k to 3 for all classic control tasks and 4 for all Atari games. In IQ-RE, we set the extra parameter d to 50 as in [32], which has been shown to be both efficient and effective. Other parameter settings can be found in Appendix C of the supplementary material. For classic control tasks, we evaluate the training performance using the average episode returns every 10k time steps for CartPole-v0 and Pendulum-v0 and 20k time steps for CartPole-v1 and Acrobot-v1. For Atari games, the time step range for performance evaluation is 200k. All experiment results reported are the average episode returns over five independent runs.

C. Baselines

We evaluate our method in comparison to the following state-of-the-art baseline methods for single-task RL.

1) DQN [5] is a representative algorithm of Deep RL, which reduces catastrophic interference using experience replay and fixed target networks. We use the DQN agent implemented in OpenAI Baselines.
2) Rainbow [44] is the upgraded version of DQN containing six extensions, including a prioritized replay buffer [27], n-step returns [2], Adam optimizer [55], and distributional learning [10] for stable RL training. The Rainbow agent is implemented in Google's Dopamine framework[7] [54].
3) SRNN [26] employs a distributional regularizer to induce sparse representations in neural networks to avert catastrophic interference in the single-task RL. Here, we implement it in the form of fully online training.
4) DSOM [25] introduces a DSOM module to control the activation of the representation output layer to achieve local optimization. We reproduce it with reference to the original DSOM implementation.[8]
5) TCNN [21] aims to remap the inputs to a high-dimensional space using tile coding to sparsify the input features, reducing activation overlap. We adopt its implementation in [56].

In the experiments, we first use DQN as the underlying RL method to evaluate the effectiveness of IQ in comparison to all baselines on classic control tasks. We then extend them to high-dimensional Atari games to further validate the scalability of IQ. Note that TCNN suffers from scalability issues for benchmarks with high dimensionality as it increases the dimension of input to the neural network, and DSOM has not been applied to solve any high-dimensional RL tasks in [25], whose implementation details are unclear. Therefore, we evaluate IQ-RE only in comparison to DQN and SRNN on the Atari games. In addition, we also implement IQ-RE with Rainbow, to illustrate that our method is highly flexible and can be incorporated into various existing value-based RL models.

[7]Dopamine is a research framework developed by Google for fast prototyping of RL algorithms: https://github.com/google/dopamine

[8]Dynamic self-organized maps: https://github.com/rougier/dynamic-som

D. Evaluation Metrics

Following the convention in previous studies [9]–[11] [24], [44], we employ the average training episode returns $\mathcal{R}_T$ to evaluate our method during training:

$$\mathcal{R}_T = \frac{1}{M}\sum_{i=1}^{M}\sum_{j=0}^{J_i} R_{ij} \tag{12}$$

where M is the number of episodes experienced within each evaluation period, J_i is the total time steps in episode i, and R_{ij} is the reward received at time step j in episode i.

E. Results

To address the effectiveness of our proposed method, we present primary results of IQ incorporated with DQN and all baselines implemented on four control tasks. Fig. 6 shows the learning curves of average episodic return during training for each task with three levels of replay buffer capacity, and Table I reports the numerical results in terms of the highest cumulative return achieved in corresponding curves. In general, IQ is clearly superior to all baselines both in terms of the stability and the maximum achieved cumulative reward, especially when the replay buffer capacity is small (e.g., $N = 100$) or even without experience replay (i.e., $N = 1$). In most tasks, IQ achieves near-optimal performance as well as good stability even without any experience replay. For Pendulum-v0 and Acrobot-v1, a large replay buffer (e.g., $N = 50\,000$) can help DQN, SRNN, and TCNN escape from catastrophic interference. However, this is not the case for two CartPole tasks where the agents exhibit fast initial learning but then encounter collapse in performance. DSOM performs comparably to IQ with large replay buffers but is significantly inferior to IQ with the other two smaller capacities. We also conduct experimental comparisons with all baselines in terms of the degree of interference [according to (5)], corresponding to all task settings in Fig. 6. The experimental results are presented in Appendix D-A of the supplementary material, from which we can further confirm that IQ can substantially reduce the negative interference encountered by the base RL agents during the learning progress.

Moreover, from a macro perspective, Fig. 6 shows that the following conditions hold.

1) DQN, SRNN, and TCNN agents exhibit high sensitivity to the replay buffer capacity. They generally perform well with a large buffer (except on CartPole-v1), but their performance deteriorates significantly when the buffer capacity is reduced. The reason for this phenomenon is that DQN primarily relies on experience replay to obtain approximately i.i.d. training data to avoid possible interference in training, which cannot be guaranteed when the replay buffer capacity is small. Since SRNN just reshapes the constraint term based on DQN loss, while TCNN only increases the input dimension of DQN, both techniques can only alleviate interference to a certain extent.
2) The overall performance of DSOM is better than the above three baselines, as it optimizes the data

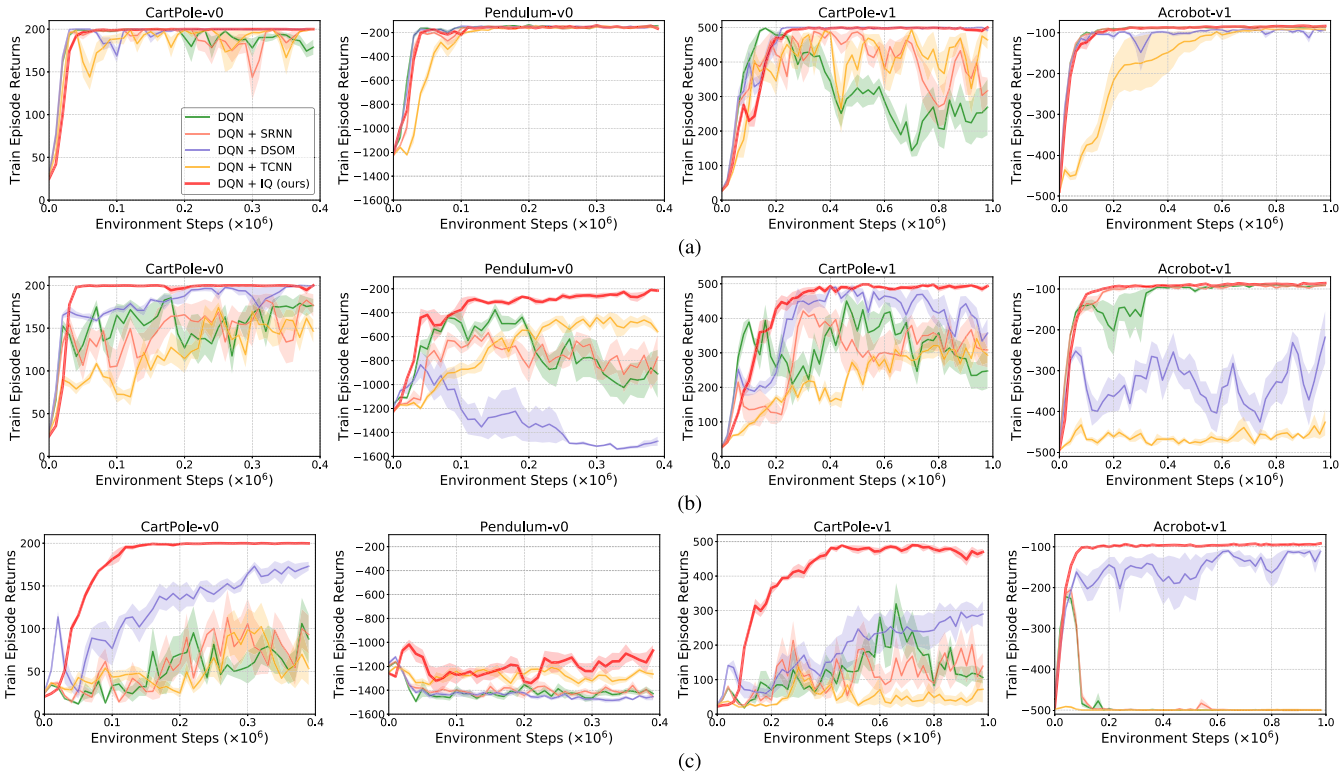


Fig. 6. Learning curves on classic control tasks with different replay buffer capacities N. Here and in related figures in the following, the solid lines and shaded regions denote the means and standard deviations of rewards, respectively, across five runs. (a) $N = 50\,000$. (b) $N = 100$. (c) $N = 1$.

TABLE I

NUMERICAL RESULTS IN TERMS OF THE HIGHEST CUMULATIVE RETURN ACHIEVED DURING TRAINING OF ALL METHODS IMPLEMENTED IN THE CLASSIC CONTROL TASKS (BASED ON THE PERFORMANCE OF FIVE RUNS IN FIG. 6. HERE AND IN RELATED TABLES, THE BEST PERFORMANCE IS MARKED IN BOLDFACE.)

Method	DQN			DQN + SRNN			DQN + DSOM			DQN + TCNN			DQN + IQ (ours)		
N	1	100	50,000	1	100	50,000	1	100	50,000	1	100	50,000	1	100	50,000
CartPole-v0	106.0	185.4	**200.0**	112.9	182.9	**200.0**	173.0	**200.0**	**200.0**	102.3	174.1	**200.0**	**200.0**	**200.0**	**200.0**
Pendulum-v0	−1165.3	−375.2	**−134.9**	−1160.3	−567.1	−144.2	−1122.1	−834.7	−146.2	−1204.8	−438.0	−142.0	**−1018.1**	**−209.1**	−140.0
CartPole-v1	188.4	421.5	497.7	250.1	445.6	498.6	295.5	490.0	500.0	136.8	341.1	494.2	**489.3**	**498.5**	**500.0**
Acrobot-v1	−222.1	−89.2	−85.4	−204.7	−89.4	−83.2	−110.0	−218.2	−91.4	−491.5	−426.5	−89.8	**−91.7**	**−85.7**	**−82.9**

distribution-specific representation modules to circumvent the interference caused by the shared representation layer. Nevertheless, since DSOM shares the same output layer, it still suffers from interference on most tasks.

3) By contrast, IQ features a shared representation module and multiple data distribution-specific output heads and employs the knowledge distillation technique to prevent interference caused by shared representation layers, achieving significantly better performance than baselines. Note that in some cases (e.g., CartPole-v1 settings), IQ learns more slowly than baselines during the early stages of training. A possible explanation is that IQ learns context division in a fully online manner and the partitions may not be accurate enough back then, but it can quickly surpass the baselines as the training progress.

In addition, to demonstrate the scalability and flexibility of our method, we also provide the results of IQ-RE with DQN and Rainbow on six Atari games. The learning curves are shown in Fig. 7 and the highest cumulative returns achieved during training are summarized in Table II. Overall, for high-dimensional image inputs, the training performance of the underlying RL algorithms can be noticeably improved with our IQ-RE scheme, while SRNN provides little contribution to both underlying RL methods. Specifically, in Fig. 7, with DQN as the underlying RL method, IQ-RE significantly outperforms DQN and SRNN on seven out of 12 tasks, being comparable with DQN and SRNN on the rest five tasks. Similarly, with Rainbow as the underlying RL method, IQ-RE outperforms baselines on eight out of 12 tasks while being comparable with baselines on the rest four tasks. The two-sample Kolmogorov–Smirnov test in Appendix D-C of the supplementary material further confirms that the improvement brought by IQ-RE is statistically significant. Furthermore, as shown in Table II, IQ-RE achieves higher maximum cumulative scores in most tasks than its counterparts. Among the 24 training settings, the maximum cumulative scores achieved by IQ-RE are slightly lower than those of baselines in only four cases

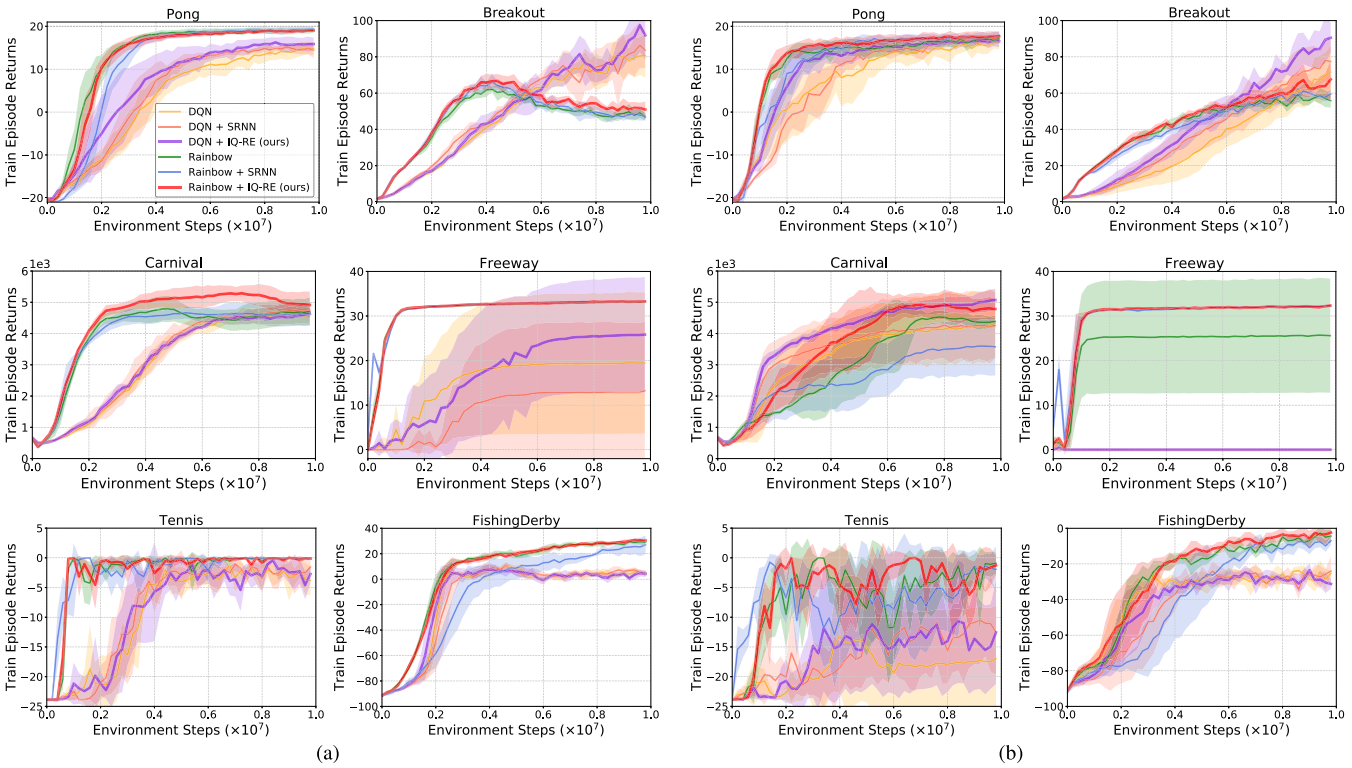


Fig. 7. Learning curves on Atari games with different replay buffer capacities N. It is worth noting that the green line and the red line of Freeway in (a) and the purple, orange, and pink lines of Freeway in (b) are overlapping. (a) $N = 1\,000\,000$. (b) $N = 10\,000$.

TABLE II

NUMERICAL RESULTS IN TERMS OF THE HIGHEST CUMULATIVE RETURN ACHIEVED DURING TRAINING OF ALL METHODS IMPLEMENTED IN THE ATARI GAMES (BASED ON THE PERFORMANCE OF FIVE RUNS IN FIG. 7)

Method	DQN		DQN + SRNN		DQN + IQ-RE (ours)		Rainbow		Rainbow + SRNN		Rainbow + IQ-RE (ours)	
N	10,000	1,000,000	10,000	1,000,000	10,000	1,000,000	10,000	1,000,000	10,000	1,000,000	10,000	1,000,000
Pong	16.2	15.0	**17.3**	15.2	16.7	**16.3**	17.1	19.1	**17.9**	**19.3**	17.7	19.0
Breakout	71.6	80.9	78.0	86.4	**90.5**	**97.4**	59.3	62.1	61.1	66.2	**67.6**	**66.7**
Carnival	4327.0	4589.0	4273.6	**4758.0**	**5073.1**	4639.8	4529.7	4788.0	3599.6	4662.2	**4928.2**	**5289.0**
Freeway	**0.5**	19.5	**0.5**	13.2	**0.5**	**25.8**	25.7	33.2	**32.3**	**33.3**	**32.3**	**33.3**
Tennis	−13.2	−1.3	**−10.5**	−1.0	−10.7	**−0.4**	**0.0**	**0.0**	−0.7	**0.0**	**0.0**	**0.0**
FishingDerby	−23.8	9.7	−24.0	**11.0**	**−23.5**	7.5	−2.7	29.4	−7.0	27.0	**−2.4**	**30.8**

and two cases when combined with DQN and Rainbow. It is worth noting that, even with large memory ($N = 1\,000\,000$), IQ-RE still shows certain advantages over the baselines.

In summary, the proposed techniques containing context division based on the clustering of all experienced states and knowledge distillation in multihead neural networks can effectively eliminate catastrophic interference caused by data drift in the single-task RL while reducing the requirement of the replay buffer capacity for off-policy RL. In addition, our method leverages a fixed randomly initialized encoder to characterize the similarity among states in the low-dimensional representation space, which can be used to partition contexts effectively for high-dimensional environments.

F. Analysis

1) Ablation Study: Since our method can be regarded as an extension to existing RL methods (e.g., DQN [5]), with three novel components (i.e., adaptive context division by online clustering, knowledge distillation, and the multihead neural network), the ablation experiments are designed as follows.

1) No clustering means using a random partition of the raw state space before learning instead of adaptive context division by online clustering.
2) No distillation means removing the distillation loss function $\mathcal{L}_{\mathcal{D}}(\theta_{\mathcal{S}}, \theta_{\mathcal{F}})$ from (11) (i.e., $\lambda = 0$).
3) No multihead means removing the context division module and optimize the neural network with a single-head output (i.e., $k = 1$). Here, the distillation term is represented as the distillation of the network before each update of the output head.

The results of ablation experiments are shown in Fig. 8 using classic control tasks for the convenience of validation. From Fig. 8, the following facts can be observed. First, across all settings, the overall performance of DQN is the worst, showing the effectiveness of the three components introduced for coping with catastrophic interference in the single-task RL, although the contribution of each component varies substan-

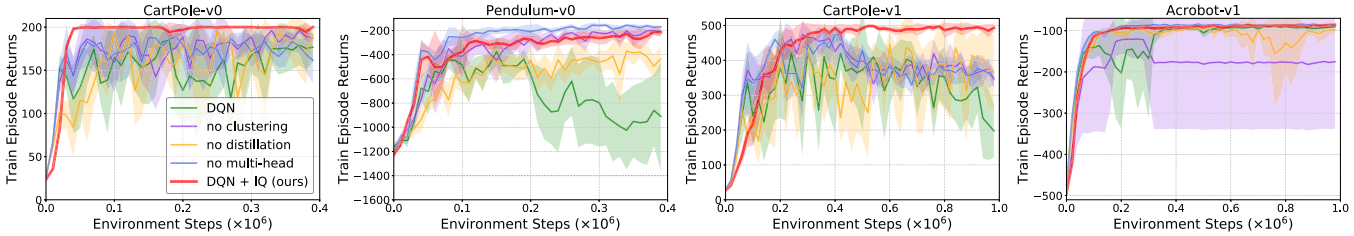


Fig. 8. Comparisons of IQ (red) with DQN (green) and its three different ablations (other colors), on each individual task ($N = 100$).

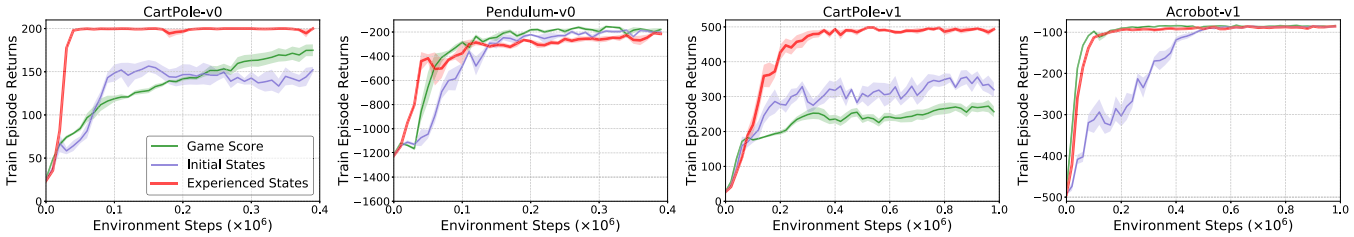


Fig. 9. Comparisons of IQ (red, it performs context division on all experienced states) with other two context division strategies (other colors) ($N = 100$).

tially per task. Second, removing online clustering from the context division module is likely to damage the performance in most cases. Third, removing knowledge distillation makes the performance deteriorate on almost all tasks, indicating that knowledge distillation is a key element in our method. Finally, without the multihead component, our model is equivalent to a DQN with an extra distillation loss, which performs better than DQN alone but worse than our proposed IQ in general. Note that, on Pendulum-v0, the single-head network performs better than our method during training, which means that the additional distillation constraint is sufficient to mitigate the interference faced by the base RL under this setting, without the need of further context division. By contrast, our method learns context division in a fully online manner and the partitions may not be accurate enough on this task. However, this is not the case in other settings (see Appendix D-E of the supplementary material for more experimental results).

2) Context Division Strategy: By introducing the context variables during learning, IQ bears a resemblance to some existing settings [47], [50] that partition contexts using the game score and initial states. Therefore, we further compare our context division based on all experienced states with the following context division strategies.

1) Game score [47] splits a game into contexts based on the undiscounted cumulative game score.
2) Initial state [50] partitions the initial state space into "slices" by the k-means clustering procedure.

From the experimental results in Fig. 9, we can observe that our method is distinctly superior to the above two strategies in general, although it performs slightly worse on Pendulum-v0. This is because, on Pendulum-v0, the game scores or initial states have a perfect correspondence to different state distributions. However, these two baselines are primarily designed to decompose complex tasks, rather than mitigating catastrophic interference in the single-task RL settings and cannot guarantee the complete decoupling among different state distributions. This conclusion is further confirmed by

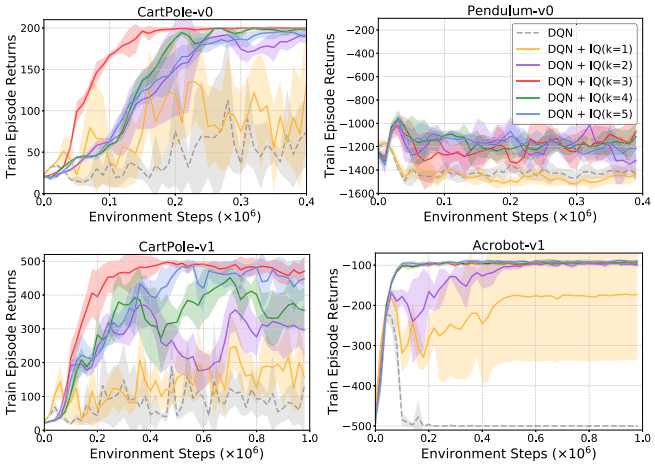
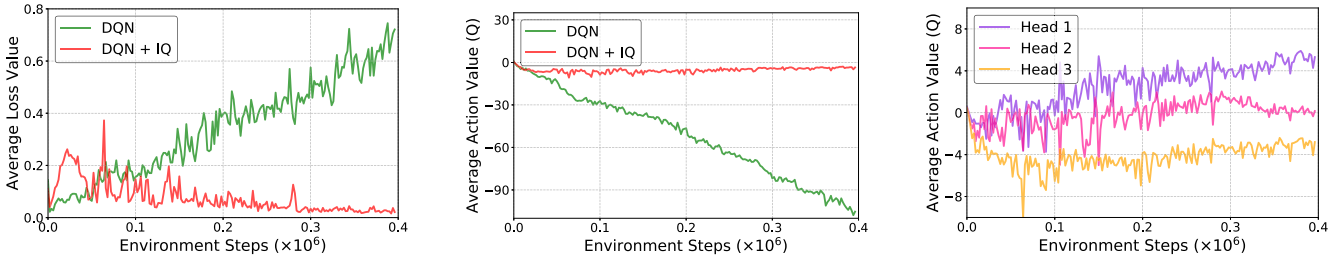


Fig. 10. Parameter sensitivity analysis with respect to the number of contexts k. Experiments are conducted with different k values ($N = 1$).

the additional experimental results in Appendix D-F of the supplementary material.

3) Parameter Analysis: There are two critical parameters in IQ: λ and k. By its nature, λ is related to the training progress. Since we need to preserve the learned good policies during training, it is intuitive to gradually increase λ until its value reaches 1. The reason is that, in early stage training, the model has not learned any sufficiently useful information, so the distillation constraint can be ignored. With the progress of training, the model starts to acquire more and more valuable information and needs to pay serious attention to interference to protect the learned good policies while learning further. In our experiments, we recommend to set λ to be inversely proportional to the exploration proportion ϵ, and the results in Figs. 6 and 7 have demonstrated the simplicity and effectiveness of this setting.

For the parameter k, it needs to be specified before training, which may be suboptimal without good knowledge of the state-space structure of the environment. To investigate the



(a) (b) (c)

Fig. 11. Training curves tracking the agent's average loss and average predicted action value for 400k environment steps in Pendulum-v0 ($N = 100$ and $k = 3$, see Fig. 6(b) for corresponding curves). (a) Each point is the average loss achieved per training iteration. (b) Average maximum predicted action value of agents on a held-out set of states. (c) Average maximum predicted action value of each output head in IQ.

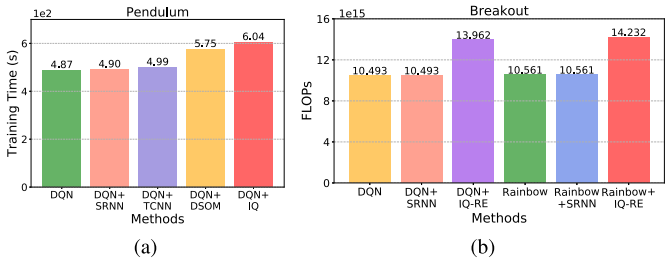
effect of k, we conduct experiments with different k values ($k \in \{1, 2, 3, 4, 5\}$) and the results are shown in Fig. 10. In our experiments, $k = 3$ is a reasonably good choice for CartPole-v0 and CartPole-v1, while $k = 5$ is best for IQ on Acrobot-v1. It is worth noting that, on Pendulum-v0, our method achieves similar performance with k set to 2, 3, 4, and 5, but without any satisfactory result. A possible explanation is that the agents failed to learn any useful information due to the limited exploration in the early training, leading to the failure of further learning.

In summary, we can make the following statements: 1) the performance of our method is obviously better than the base RL baseline regardless of the specific k value, confirming the effectiveness of IQ even with inaccurate k estimation, and 2) for $k > 1$, better performance of IQ can be expected. However, large k values are not always desirable as it will result in more fine-grained context divisions and more complex neural networks with a large amount of output heads, making the model unlikely to converge satisfactorily within a limited number of training steps. Thus, we recommend to set the value of k by taking into consideration the state-space structure of specific tasks. In practice, we recommend to explore the environment using an appropriate random policy and conduct initial density-based clustering for the obtained states before training. Thereafter, the initial centroids of SKM and k value can be estimated according to this initial clustering result.

4) Convergence Analysis: To analyze convergence, we track the agent's average loss and average predicted action value during the training progress. According to Fig. 11, we can conclude that: 1) our method has better convergence and stability in face of interference compared with original RL algorithms [see Fig. 11(a) and (b)] and 2) for a held-out set of states,[9] the average maximum predicted action value of each output head reflects the difference as expected [see Fig. 11(c)], and the final output of IQ is synthesized based on all of them.

5) Computational Efficiency: Our methods greatly improve the training performance of the existing RL algorithms, which are computationally efficient in which the following conditions hold.

[9]It refers to a fixed set of states [4], [5] of the environment, which can be obtained by performing exploration in the environment and then used to track the average predicted action value changes during training.



(a) (b)

Fig. 12. Comparison of computational efficiency. (a) Training time of each agent to achieve its performance for 400k environment steps in Pendulum-v0 ($N = 100$, see Fig. 6(b) for corresponding learning curves). (b) Number of FLOPs used by each agent at 10M environment steps in Breakout (computational complexity). Here, we only consider forward and backward passes through neural network layers (see Fig. 7 for corresponding learning curves).

1) In each time step, the extra context division module only needs to compute the distances between the current state and k context centroids, which is computationally negligible with respect to the SGD complexity of the large parameter vector updated in each iteration of RL itself.
2) Only $k - 1$ extra output heads are added to the neural network, in which the increased computation is acceptable with respect to the representation complexity.
3) There are no gradient updates through the random encoder.
4) There is no unnecessary distance computation for finding the corresponding context at every update step as the context label for each state is stored in the replay buffer.

Fig. 12 shows the training time of each agent on Pendulum and the floating-point operations (FLOPs) executed by agents on Breakout.

VI. Conclusion and Future Work

In this article, we propose a competent scheme IQ to tackle the inherent challenge of catastrophic interference in the single-task RL. The core idea is to partition all states experienced during training into a set of contexts using online clustering techniques and simultaneously estimate the context-specific value function with a multihead neural network as well as a knowledge distillation loss to mitigate the interference across contexts. Furthermore, we introduced a random

encoder to enhance the context division for high-dimensional complex tasks. Our method can effectively decouple the correlations among differently distributed states and can be easily incorporated into various value-based RL models. Experiments on several benchmarks show that our method can significantly outperform state-of-the-art RL methods and dramatically reduce the memory requirement of existing RL methods.

In the future, we aim to incorporate our method into policy-based RL models to reduce the interference during training by applying weight or functional regularization on policies. Furthermore, we will investigate a more challenging setting called continual RL in nonstationary environments [46]. This setting is a more realistic representation of the real-world scenarios and includes abrupt changes or smooth transitions on dynamics or even the dynamics itself is shuffled.

REFERENCES

[1] 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.

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

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

[4] V. Mnih *et al.*, "Playing Atari with deep reinforcement learning," in *Proc. 27th Conf. Neural Inf. Process. Syst.*, 2013, pp. 1–9.

[5] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[6] A. Faust *et al.*, "PRM-RL: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 5113–5120.

[7] H.-T.-L. Chiang, A. Faust, M. Fiser, and A. Francis, "Learning navigation behaviors end-to-end with AutoRL," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 2007–2014, Apr. 2019.

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

[9] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. 33rd Int. Conf. Mach. Learn.*, vol. 48, Jun. 2016, pp. 1928–1937.

[10] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2017, pp. 449–458.

[11] L. Espeholt *et al.*, "IMPALA: Scalable distributed Deep-RL with importance weighted actor-learner architectures," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 1407–1416.

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

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

[14] C. Fernando *et al.*, "PathNet: Evolution channels gradient descent in super neural networks," 2017, *arXiv:1701.08734*.

[15] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *Proc. 31st Conf. Neural Inf. Process. Syst.*, 2017, pp. 6467–6476.

[16] S.-A. 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. 2001–2010.

[17] 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.

[18] M. Delange *et al.*, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Feb. 5, 2021, doi: 10.1109/TPAMI.2021.3057446.

[19] S. Kessler, J. Parker-Holder, P. Ball, S. Zohren, and S. J. Roberts, "UNCLEAR: A straightforward method for continual reinforcement learning," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 1–9.

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

[21] S. Ghiassian, B. Rafiee, Y. L. Lo, and A. White, "Improving performance in reinforcement learning by breaking generalization in neural networks," in *Proc. 19th Int. Conf. Auton. Agents Multiagent Syst.*, 2020, pp. 1–10.

[22] E. Bengio, J. Pineau, and D. Precup, "Interference and generalization in temporal difference learning," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 767–777.

[23] T. Schaul, D. Borsa, J. Modayil, and R. Pascanu, "Ray interference: A source of plateaus in deep reinforcement learning," 2019, *arXiv:1904.11455*.

[24] W. Fedus, D. Ghosh, J. D. Martin, M. G. Bellemare, Y. Bengio, and H. Larochelle, "On catastrophic interference in Atari 2600 games," 2020, *arXiv:2002.12499*.

[25] Y. L. Lo and S. Ghiassian, "Overcoming catastrophic interference in online reinforcement learning with dynamic self-organizing maps," in *Proc. 33rd Conf. Neural Inf. Process. Syst.*, 2019, pp. 1–9.

[26] V. Liu, R. Kumaraswamy, L. Le, and M. White, "The utility of sparse representations for control in reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, no. 1, pp. 4384–4391.

[27] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. 4th Int. Conf. Learn. Represent.*, 2016, pp. 1–21.

[28] W. Fedus *et al.*, "Revisiting fundamentals of experience replay," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 3061–3071.

[29] J. G. Dias and M. J. Cortinhal, "The SKM algorithm: A K-means algorithm for clustering sequential data," in *Proc. Ibero-Amer. Conf. Artif. Intell.*, 2008, pp. 173–182.

[30] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 3987–3995.

[31] S. Golkar, M. Kagan, and K. Cho, "Continual learning via neural pruning," 2019, *arXiv:1903.04476*.

[32] Y. Seo, L. Chen, J. Shin, H. Lee, P. Abbeel, and K. Lee, "State entropy maximization with random encoders for efficient exploration," 2021, *arXiv:2102.09430*.

[33] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *J. Artif. Intell. Res.*, vol. 47, pp. 253–279, Jun. 2013.

[34] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. D. Rodríguez, "Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges," *Inf. Fusion*, vol. 58, pp. 52–68, Dec. 2020.

[35] D. Isele and A. Cosgun, "Selective experience replay for lifelong learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1, pp. 1–9.

[36] D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap, and G. Wayne, "Experience replay for continual learning," in *Proc. 33th Conf. Neural Inf. Process. Syst.*, 2019, pp. 1–11.

[37] 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.

[38] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *Proc. Workshop Conf. Neural Inf. Process. Syst.*, 2015, pp. 1–9.

[39] A. A. Rusu *et al.*, "Policy distillation," in *Proc. 5th Int. Conf. Learn. Represent.*, 2016, pp. 1–31.

[40] T. Gangwani and J. Peng, "Policy optimization by genetic distillation," in *Proc. 7th Int. Conf. Learn. Represent.*, 2018, pp. 1–16.

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

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

[43] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, vol. 30, no. 1, pp. 1–7.

[44] M. Hessel *et al.*, "Rainbow: Combining improvements in deep reinforcement learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1, pp. 1–8.

[45] S. Padakandla, K. J. Prabuchandran, and S. Bhatnagar, "Reinforcement learning algorithm for non-stationary environments," *Appl. Intell.*, vol. 50, no. 11, pp. 3590–3606, Jun. 2020.

[46] 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. 248–249.
[47] V. Jain, W. Fedus, H. Larochelle, D. Precup, and M. G. Bellemare, "Algorithmic improvements for deep reinforcement learning applied to interactive fiction," in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 4, pp. 4328–4336.
[48] K. Milan, J. Veness, J. Kirkpatrick, M. Bowling, A. Koop, and D. Hassabis, "The forget-me-not process," in *Proc. 30th Conf. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 3702–3710.
[49] D. Rao, F. Visin, A. A. Rusu, Y. W. Teh, R. Pascanu, and R. Hadsell, "Continual unsupervised representation learning," in *Proc. 33th Conf. Neural Inf. Process. Syst.*, 2019, pp. 1–11.
[50] D. Ghosh, A. Singh, A. Rajeswaran, V. Kumar, and S. Levine, "Divide-and-conquer reinforcement learning," in *Proc. 7th Int. Conf. Learn. Represent.*, 2018, pp. 1–14.
[51] 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.
[52] V. Liu, A. White, H. Yao, and M. White, "Towards a practical measure of interference for reinforcement learning," 2020, *arXiv:2007.03807*.
[53] G. Brockman *et al.*, "OpenAI Gym," 2016, *arXiv:1606.01540*.
[54] P. S. Castro, S. Moitra, C. Gelada, S. Kumar, and M. G. Bellemare, "Dopamine: A research framework for deep reinforcement learning," 2018, *arXiv:1812.06110*.
[55] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, 2014, pp. 1–15.
[56] Y. Pan, K. Banman, and M. White, "Fuzzy tiling activations: A simple approach to learning sparse representations online," in *Proc. 9th Int. Conf. Learn. Represent.*, 2021, pp. 1–30.

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.

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 his positions as a Post-Doctoral Researcher, an Associate Researcher, and a Researcher with the China Academy of Space Technology (CAST), Beijing. From 2003 to 2007, he held his positions as a Researcher and an 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. He is currently an Associate Professor with the Division of Informatics, Shenzhen International Graduate School, Tsinghua University, Shenzhen, China. 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.