Efficient Distributional Reinforcement Learning with Kullback-Leibler Divergence Regularization

Renxing Li, Zhiwei Shang, Chunhua Zheng, Huiyun Li, Qing Liang, Yunduan Cui

***Abstract*—In this article, we address the issues of stability and data-efficiency in reinforcement learning (RL). A novel RL approach, Kullback–Leibler divergence-regularized distributional RL (KLC51) is proposed to integrate the advantages of both stability in the distributional RL and data-efficiency in the Kullback-Leibler (KL) divergence-regularized RL in one framework. KLC51 derived the Bellman equation and the TD errors regularized by KL divergence in a distributional perspective and explored the approximated strategies of properly mapping the corresponding Boltzmann softmax term into distributions. Evaluated by several benchmark tasks with different complexity, the proposed method clearly illustrates the positive effect of the KL divergence regularization to the distributional RL including exclusive exploration behaviors and smooth value function update, and successfully demonstrates its significant superiority in both learning stability and data-efficiency compared with the related baseline approaches.**

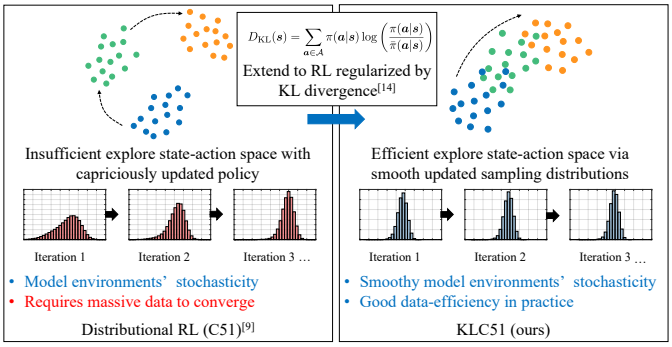

Fig. 1. Principle of the proposed approach in this paper.

I. Introduction

REINFORCEMENT Learning (RL) [1] is one of the most exciting fields of Machine Learning. Driving agents to learn optimal or suboptimal control policies from unknown environments through a trail-and-error mechanism, it provides an appealing prospect of learning task without human knowledge and produces many interesting applications over the years, particularly in robot and autonomous control problems [2]–[4]. Supported by the great nonlinear approximation capability of deep neural networks, deep reinforcement learning (DRL) [5] has surpassed human in a wide range of challenging games designed by code or human-made rule [6], [7] once the agents were sufficiently trained to extract the informative features from the complicated state-action space. These results clearly demonstrated its potential as an emerging direction of artificial intelligence. On the other hand, the practical DRL in the real-world control problems still remains limited due to not only the noisy environment but also its data-driven nature. The conventional RL approach models the external environments without intrinsic stochasticity while the noise and disturbance in the real-world scenario could easily result in an unstable and deteriorated learning procedure with overestimation of the value function [8]. Further more, DRL approach usually requires exploring a huge number of samples to sufficiently train its networks over the high-dimensional state-action space which is extremely expensive in the real-world hardwares like robots [9], [10].

To tackle the issue of the unstable learning caused by the intrinsic stochasticity in environments, the distributional RL (C51) [11] was proposed to describe the cumulative reward in value function as a distribution rather than the expected value in the conventional methods. With the superiority of gaining more informative knowledge about the stochastic environment and learning better representations to avoid state aliasing, the distributional RL successfully learned a set of auxiliary tasks tightly coupled to the reward as more robust learning targets [12]. It not only outperformed the conventional approaches in various Atari simulation tasks with less prediction variance [11], but also demonstrated good poetntal in various engineering scenarios including power system [13] and autonomous driving [14]. According to the theoretic study in [15], [16], the distributional RL also improves the learning quality of its deep neural networks structure, especially under noisy environments. As one solution to the data-efficiency of RL, the dynamic policy programming (DPP) [17] was proposed by introducing Kullback–Leibler (KL) divergence between current and new policies as a penalty term in its

This work is supported in part by the National Natural Science Foundation of China under Grant 62103403; in part by the National Key Research and Development Program of China under Grant 2020YFB2104300; in part by Department of Science and Technology of Guangdong Province under Grant 2021A0505030056; in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2020B515130004; in part by the Science and Technology Development Fun, Macao S.A.R. (FDCT) under Grant 0015/2019/AKP.

Corresponding author: Yunduan Cui (email: yd.cui@siat.ac.cn)

Renxing Li is with School of Software Engineering, University of Science and Technology of China, China.

Zhiwei Shang is with Shenzhen Institute of Advanced Technology (SIAT), Chinese Academy of Sciences, China and University of Chinese Academy of Sciences, China.

Qing Liang is with Department of Automation, University of Science and Technology of China, China.

Chunhua Zheng, Huiyun Li and Yunduan Cui are with CAS Key Laboratory of Human-Machine Intelligence-Synergy Systems, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, and Guangdong-Hong Kong-Macao Joint Laboratory of Human-Machine Intelligence-Synergy Systems, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, Guangdong, 518055, China.

TABLE I
COMPARISON OF KLC51 AND THE RELATED APPROACHES

Approach	Distributional	KL regularized
DQN [6]	×	×
C51 [11]	◯	×
DPP [17]	×	◯
KLC51 (ours)	◯	◯

value function to avoid the over-large policy update. Such an RL framework regularized by KL divergence is theoretically proved to have the state-of-the-art error dependency as it implicitly averages over all previous action value functions and hence also averages errors according to [18], [19]. This characteristic contributed to the great data-efficiency in various challenging engineering applications from robot manipulation [20], [21] to chemical plant control [22], [23] where the agents quickly explored the task within limited number of interactions through smoothly updated policies.

This paper aims to address both the stability and the data-efficiency issues in RL. Following the principle described in Fig. 1, the traditional distributional RL [11] stablizes the learning procedure by modeling the intrinsic stochasticity of environments without tackling the issue of data-efficiency. Its policy may be capriciously updated without constraint and turns to an insufficient exploration over high-dimensional state-action space. We proposed a novel approach, KL divergence-regularized distributional RL (KLC51) as an extension of distributional RL [11] to the RL regularized by KL divergence [17], [19]. With a smooth update between the current and previous policies, it efficiently explored the state-action space in a distributional perspective. According to the relationship between the proposed method and other related works concluded in Table I, KLC51 integrates the advantages of C51 [11] and DPP [17], [19] in one framework. It extended the distributional forms of C51 towards the KL divergence-regularized Bellman equation and the temporal difference (TD) error and novelly derived an approximated update strategy to properly map the Boltzmann softmax operator of DPP into distributions. Evaluated by several benchmark control tasks, the proposed KLC51 naturally integrated the stability of C51 and the data-efficiency of DPP in one algorithm and successfully demonstrated a superior learning performance compared with other baseline approaches. The contributions of this paper are summarized as:

1) For the theoretic study, this work novelly derived the Bellman equation and TD errors of the KL divergence-regularized RL in a distributional perspective.
2) For the algorithm development, the proposed KLC51 successfully integrated both the stability of C51 against noisy environments and the data-efficiency of DPP in engineering applications within one algorithm.
3) For the experimental validations, the proposed algorithm was evaluated in various control problems to study its learning capability and data-efficiency compared with the related baseline approaches. The positive effects of distributional value function and the KL divergence were further demonstrated via data visualization.

The remainder of this paper is organized as follows. The preliminaries of RL was in Section II. Section III detailed the proposed KLC51. The experimental results were demonstrated in Section IV. Section V presented the conclusion.

II. PRELIMINARIES

A. Expected Reinforcement Learning

The conventional RL models the environments as a Markov decision process (MDP) with a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. For a data set with n samples, $\mathcal{S} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, ..., \boldsymbol{s}_n\}$ and $\mathcal{A} = \{\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_n\}$ are the finite sets of state and action. $\mathcal{P}^{\boldsymbol{a}}_{\boldsymbol{ss}'} = p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$ represents the transition probability of switching from state $\boldsymbol{s}$ to $\boldsymbol{s}'$ under action $\boldsymbol{a}$. $\mathcal{R}$ is the reward function, $\mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{ss}'}$ represents the immediate reward obtained by the transition above. The discount factor $\gamma \in [0, 1]$ exponentially discounts the future rewards. A policy control $\pi(\boldsymbol{a}|\boldsymbol{s})$ maps the state $\boldsymbol{s}$ to a probability distribution over the action $\boldsymbol{a}$.

The value function in the expected RL is defined as the expectation of long-term return from initial state $\boldsymbol{s}_0$, under the policy π:

$$V_\pi(\boldsymbol{s}) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_{\boldsymbol{s}_t} | \boldsymbol{s}_0 = \boldsymbol{s} \right], \tag{1}$$

where $r_{\boldsymbol{s}_t} = \sum_{\substack{\boldsymbol{a} \in \mathcal{A} \\ \boldsymbol{s}_{t+1} \in \mathcal{S}}} \pi(\boldsymbol{a}|\boldsymbol{s}_t) \mathcal{P}^{\boldsymbol{a}}_{\boldsymbol{s}_t \boldsymbol{s}_{t+1}} \mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{s}_t \boldsymbol{s}_{t+1}}$ is the expected reward from state $\boldsymbol{s}_t$ over the action set $\mathcal{A}$. The optimal value funtion that maximizes the expected discounted total reward from any initial state follows a Bellman equation:

$$V^*(\boldsymbol{s}) = \max_\pi \sum_{\substack{\boldsymbol{a} \in \mathcal{A} \\ \boldsymbol{s}' \in \mathcal{S}}} \pi(\boldsymbol{a}|\boldsymbol{s}) \mathcal{P}^{\boldsymbol{a}}_{\boldsymbol{ss}'} \left(\mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{ss}'} + \gamma V^*(\boldsymbol{s}') \right) \tag{2}$$

where the optimal policy π^* follows:

$$\pi^* = \arg\max_\pi \sum_{\substack{\boldsymbol{a} \in \mathcal{A} \\ \boldsymbol{s}' \in \mathcal{S}}} \pi(\boldsymbol{a}|\boldsymbol{s}) \mathcal{P}^{\boldsymbol{a}}_{\boldsymbol{ss}'} (\mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{ss}'} + \gamma V^*(\boldsymbol{s}')). \tag{3}$$

It is also convenient to describe the expectation of long-term return of each state-action pair under the policy π by the Q function:

$$Q_\pi(\boldsymbol{s}, \boldsymbol{a}) = \sum_{\boldsymbol{s}' \in \mathcal{S}} \mathcal{P}^{\boldsymbol{a}}_{\boldsymbol{ss}'} (\mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{ss}'} + \gamma \sum_{\boldsymbol{a}' \in \mathcal{A}} \pi(\boldsymbol{a}'|\boldsymbol{s}') Q_\pi(\boldsymbol{s}', \boldsymbol{a}')), \tag{4}$$

The expected RL like DQN [6] aims to approximate the Q function by deep neural networks. Define the network parameters in iteration i as $\boldsymbol{\theta}_i$,the approximated Q function follows:

$$\hat{Q}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}_i) = \mathbb{E}[\mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{ss}'} + \gamma \max_{\boldsymbol{a}'} \hat{Q}(\boldsymbol{s}, \boldsymbol{a}', \boldsymbol{\theta}_{i-1})]. \tag{5}$$

The parameters are updated by minimizing the mean squared TD error via gradient based optimization approaches with learning rate α:

$$L(\boldsymbol{\theta}_i) = \mathbb{E}[\mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{ss}'} + \gamma \max_{\boldsymbol{a}'} \hat{Q}(\boldsymbol{s}', \boldsymbol{a}', \boldsymbol{\theta}_{i-1}) - \hat{Q}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}_i)]^2. \tag{6}$$

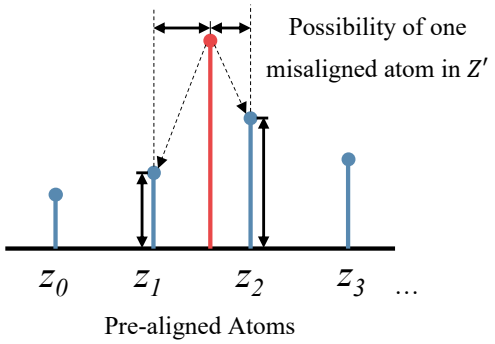

Fig. 2. Probabilities of the misaligned atoms in Z' are splited into the closest pre-aligned atoms of z_i based on the weight of their distances.

In practice, two networks are built in DQN to stablize the training: the main network $\hat{Q}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta})$ predicts $Q(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta}_i)$, and a target network $\hat{Q}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta}^-)$ with frozen parameters to estimate $\hat{Q}(\boldsymbol{s}',\boldsymbol{a}',\boldsymbol{\theta}_{i-1})$. The target network copies the parameters from the main network $\boldsymbol{\theta}^- = \boldsymbol{\theta}$ every C step.

B. Distributional Reinforcement Learning

The distributional RL [11] is proposed to address the intrinsic randomness in the long-term return by modeling the full distribution of the total future rewards, instead of its expectation. A distributional Bellman equation of a random return Z whose expectation is Q is defined as:

$$Z_\pi(\boldsymbol{s},\boldsymbol{a}) \overset{D}{=} \mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{ss}'} + \gamma Z_\pi(\boldsymbol{s}',\boldsymbol{a}'), \tag{7}$$

where $U \overset{D}{=} V$ denotes that the random variable U has the same distribution of V. Similar to the Q function, Z function can be updated based on the TD errors. Calculate the target value distribution $Z'_\pi(\boldsymbol{s},\boldsymbol{a})$ as:

$$Z'_\pi(\boldsymbol{s},\boldsymbol{a}) \overset{D}{=} \mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{ss}'} + \gamma Z_\pi(\boldsymbol{s}',\boldsymbol{a}^*), \tag{8}$$

$$\boldsymbol{a}^* = \arg\max_{\boldsymbol{a}'} \mathbb{E}\left[Z(\boldsymbol{s}',\boldsymbol{a}')\right] = \arg\max_{\boldsymbol{a}'} Q(\boldsymbol{s}',\boldsymbol{a}'). \tag{9}$$

The TD error is defined as the distance between $Z(\boldsymbol{s},\boldsymbol{a})$ and $Z'(\boldsymbol{s},\boldsymbol{a})$. The corresponding optiml policy of $Z(\boldsymbol{s},\boldsymbol{a})$ follows:

$$\pi^* = \arg\max_{\pi} \sum_{\boldsymbol{a}\in\mathcal{A}} \pi(\boldsymbol{a}|\boldsymbol{s}) \mathbb{E}\left[Z(\boldsymbol{s},\boldsymbol{a})\right]. \tag{10}$$

As the most well known distributional RL approach in practice, C51 [11] employs a discrete distribution parametrized by $N \in \mathbb{N}$ and $V_{MIN}, V_{MAX} \in \mathbb{R}$ to model $Z_\pi(\boldsymbol{s},\boldsymbol{a})$ as a set of atoms:

$$z_i = V_{MIN} + i\Delta z, \quad 0 \le i < N \tag{11}$$

where the interval $\Delta z = \frac{V_{MAX}-V_{MIN}}{N-1}$. Set $N = 51$, these atoms represents the canonical returns of the distribution $Z_\pi(\boldsymbol{s},\boldsymbol{a})$ with the corresponding probabilities $p(\boldsymbol{s},z_i,\boldsymbol{a})$. Such a discrete distribution has proved to be highly expressive and computationally friendly to describe arbitrary distributions [24]. After processing the misalignment between target distribution Z' and the pre-aligned atoms z_i follwoing Fig. 2, KL divergence is utilized to quantify the TD error between the current and target distributions which is reduced via gradient based optimization approaches on neural networks following other DRL methods [6]. The resulted Q fucntion can be recovered following:

$$Q(\boldsymbol{s},\boldsymbol{a}) = \mathbb{E}\left[Z(\boldsymbol{s},\boldsymbol{a})\right] = \sum_{i=0}^{N-1} p(\boldsymbol{s},z_i,\boldsymbol{a}) z_i. \tag{12}$$

III. Approach

A. Distributional Bellman Equation with Kullback-Leibler Divergence

In this subsection, we derived the KL divergence-regularized RL in a distributional perspective. The proposed method, KLC51 took inspiration from DPP [17] which improves data-efficiency by considering the KL divergence between current and new policies as a regularization term while inheriting the distributional features from C51 [11].

DPP introduces the KL divergence between the current policy $\pi(\boldsymbol{a}|\boldsymbol{s})$ and a baseline policy $\bar{\pi}(\boldsymbol{a}|\boldsymbol{s})$ to its value function as a regularization term:

$$V^\pi_{\bar{\pi}}(\boldsymbol{s}) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t \left(r_{\boldsymbol{s}_t} - \frac{1}{\eta} D_{\mathrm{KL}}\left(\boldsymbol{s}_t\right)\right) \mid \boldsymbol{s}_0 = \boldsymbol{s}\right], \tag{13}$$

$$D_{\mathrm{KL}}(\boldsymbol{s}) = \sum_{\boldsymbol{a}\in\mathcal{A}} \pi(\boldsymbol{a}|\boldsymbol{s}) \log\left(\frac{\pi(\boldsymbol{a}|\boldsymbol{s})}{\bar{\pi}(\boldsymbol{a}|\boldsymbol{s})}\right) \tag{14}$$

where $\eta \in (0,1]$ controls the effect of the Kullback-Leibler divergence term. Introduce Eq. (13) into Eq. (2), the corresponding optimal value function also satisfies a Bellman equation:

$$V^*(\boldsymbol{s}) = \max_\pi \sum_{\substack{\boldsymbol{a}\in\mathcal{A}\\ \boldsymbol{s}'\in\mathcal{S}}} \pi(\boldsymbol{a}|\boldsymbol{s}) \mathcal{P}^{\boldsymbol{a}}_{\boldsymbol{ss}'} \left(\mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{ss}'} + \gamma V^*(\boldsymbol{s}')\right) - \frac{1}{\eta} D_{\mathrm{KL}}(\boldsymbol{s}). \tag{15}$$

Following [17], [25], the optimal value function and policy can be calculated in the following loop at the t-th iteration:

$$V^\pi_{\bar{\pi},t+1}(\boldsymbol{s}) = \frac{1}{\eta} \log \sum_{\boldsymbol{a}\in\mathcal{A}} \bar{\pi}_t(\boldsymbol{a}|\boldsymbol{s}) \exp\left(\eta \sum_{\boldsymbol{s}'\in\mathcal{S}} \mathcal{P}^{\boldsymbol{a}}_{\boldsymbol{ss}'} \left(\mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{ss}'} + \gamma V^\pi_{\bar{\pi},t}(\boldsymbol{s}')\right)\right), \tag{16}$$

$$\bar{\pi}_{t+1}(\boldsymbol{a}|\boldsymbol{s}) = \frac{\bar{\pi}_t(\boldsymbol{a}|\boldsymbol{s}) \exp\left[\eta \sum_{\boldsymbol{s}'\in\mathcal{S}} \mathcal{P}^{\boldsymbol{a}}_{\boldsymbol{ss}'} \left(\mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{ss}'} + \gamma V^\pi_{\bar{\pi},t}(\boldsymbol{s}')\right)\right]}{\exp\left(\eta V^\pi_{\bar{\pi},t+1}(\boldsymbol{s})\right)}. \tag{17}$$

Such a loop can be further simplified based on the action preferences function [1] Ψ which is close to the Q function:

$$\Psi_{t+1}(\boldsymbol{s},\boldsymbol{a}) = \frac{1}{\eta} \log \bar{\pi}_t(\boldsymbol{a}|\boldsymbol{s}) + \sum_{\boldsymbol{s}'\in\mathcal{S}} \mathcal{P}^{\boldsymbol{a}}_{\boldsymbol{ss}'} \left(\mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{ss}'} + \gamma V^\pi_{\bar{\pi},t}(\boldsymbol{s}')\right). \tag{18}$$

The simplified update loop becomes:

$$V^\pi_{\bar{\pi},t+1}(\boldsymbol{s}) = \frac{1}{\eta} \log \sum_{\boldsymbol{a}\in\mathcal{A}} \exp\left(\eta \Psi_t(\boldsymbol{s},\boldsymbol{a})\right), \tag{19}$$

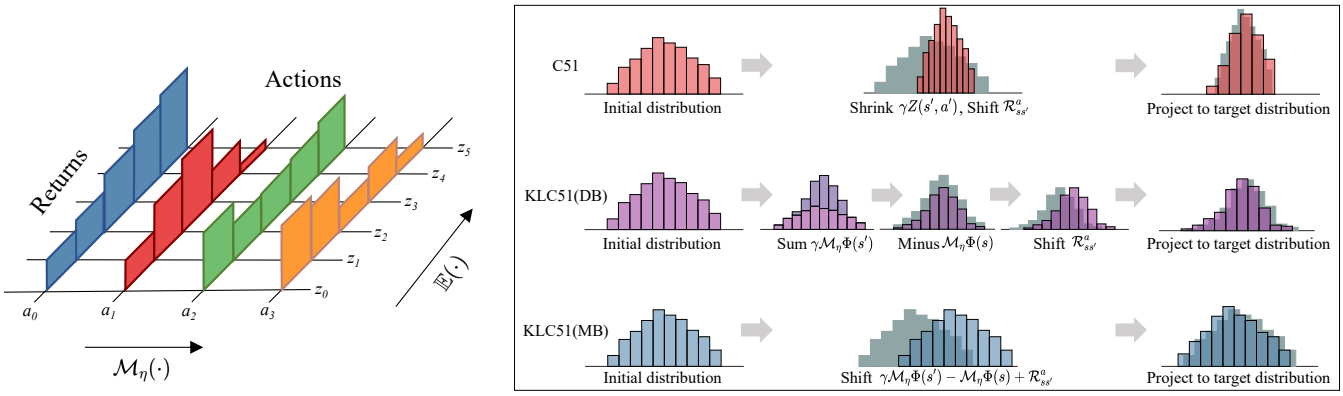

Fig. 3. Letf: Boltzmann softmax operator of KLC51 in a distributional perspective; right: different strategies of calculating distributional TD errors in KLC51.

$$\bar{\pi}_{t+1}(\boldsymbol{a}|\boldsymbol{s}) = \frac{\exp(\eta\Psi_t(\boldsymbol{s},\boldsymbol{a}))}{\sum_{\boldsymbol{a}'\in\mathcal{A}}\exp(\eta\Psi_t(\boldsymbol{s},\boldsymbol{a}'))}. \tag{20}$$

The action preferences function is updated by pluging the loop above into Eq. (18):

$$\Psi_{t+1}(\boldsymbol{s},\boldsymbol{a}) = \Psi_t(\boldsymbol{s},\boldsymbol{a}) - \mathcal{L}_\eta\Psi_t(\boldsymbol{s}) + \sum_{\boldsymbol{s}'\in\mathcal{S}}\mathcal{P}^{\boldsymbol{a}}_{\boldsymbol{ss}'}(\mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{ss}'} + \gamma\mathcal{L}_\eta\Psi_t(\boldsymbol{s}')), \tag{21}$$

$$\mathcal{L}_\eta\Psi(\boldsymbol{s}) \triangleq \frac{1}{\eta}\log\sum_{\boldsymbol{a}\in\mathcal{A}}\exp(\eta\Psi(\boldsymbol{s},\boldsymbol{a})). \tag{22}$$

In this work, we extended the distributional form of action preferences function Ψ as Φ which satisfies:

$$\Psi_{t+1}(\boldsymbol{s},\boldsymbol{a}) = \mathbb{E}\left[\Phi_{t+1}(\boldsymbol{s},\boldsymbol{a})\right]. \tag{23}$$

The update rule of Φ is derived following Eq. (22) without the expectation over the next step state $\boldsymbol{s}'$:

$$\Phi_{t+1}(\boldsymbol{s},\boldsymbol{a}) \overset{D}{=} \Phi_t(\boldsymbol{s},\boldsymbol{a}) - \mathcal{L}_\eta\Phi_t(\boldsymbol{s}) + \mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{ss}'} + \gamma\mathcal{L}_\eta\Phi_t(\boldsymbol{s}'). \tag{24}$$

According to [17], the operator $\mathcal{L}_\eta$ could be replaced to the Boltzmann softmax operator for a more analytically tractable recursion:

$$\mathcal{M}_\eta\Phi(\boldsymbol{s}) = \sum_{\boldsymbol{a}\in\mathcal{A}}\frac{\exp(\eta\Phi(\boldsymbol{s},\boldsymbol{a}))\Phi(\boldsymbol{s},\boldsymbol{a})}{\sum_{\boldsymbol{a}'\in\mathcal{A}}\exp(\eta\Phi(\boldsymbol{s},\boldsymbol{a}'))}. \tag{25}$$

The final update rule of the distributional action preferences function Φ therefore satisfies:

$$\Phi_{t+1}(\boldsymbol{s},\boldsymbol{a}) \overset{D}{=} \Phi_t(\boldsymbol{s},\boldsymbol{a}) - \mathcal{M}_\eta\Phi_t(\boldsymbol{s}) + \mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{ss}'} + \gamma\mathcal{M}_\eta\Phi_t(\boldsymbol{s}'). \tag{26}$$

B. Distributional TD Error with Kullback-Leibler Divergence

In this subsection, we derived the TD errors in Eq. (26). Compared with the one of C51 in Eq. (8) where $Z_\pi(\boldsymbol{s}',\boldsymbol{a}^*)$ is calculated based on a greedy action $\boldsymbol{a}^*$, it requires calculating the Boltzmann softmax terms $-\mathcal{M}_\eta\Phi(\boldsymbol{s})$ and $\gamma\mathcal{M}_\eta\Phi(\boldsymbol{s}')$ over the dimension of actions as described in the left side of Fig. 3. How to properly process the Boltzmann softmax operator $\mathcal{M}_\eta(\cdot)$ to the distributional actions becomes the key of the proposed approach. Following the illustration in the right side of Fig. 3, C51 calculates the target distribution of action preferences function in the following step: 1) shrink the initial distribution with the greedy action $Z(\boldsymbol{s}',\boldsymbol{a}^*)$ by the discount factor γ; 2) shift the distribution by reward $\mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{ss}'}$; 3) project the distribution to the pre-aligned 51 atoms following Section II-B.

We proposed two strategies to calculate the Boltzmann softmax terms in a distributional perspective. The first strategy is distribution of Boltzmann softmax (DB) which directly calculates the Boltzmann softmax terms as distributions. Given an initial distribution $\Phi(\boldsymbol{s},\boldsymbol{a})$, KLC51 (DB) first adds it with a shrinked Boltzmann softmax term $\gamma\mathcal{M}_\eta\Phi(\boldsymbol{s}')$ and then minus another term $\mathcal{M}_\eta\Phi(\boldsymbol{s})$. The resulted distribution is finally shifted by reward $\mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{ss}'}$ and projected to the pre-aligned atoms following C51. This strategy enjoys the full distribution information on the Boltzmann softmax operator. On the other hand, since both $\gamma\mathcal{M}_\eta\Phi(\boldsymbol{s}')$ and $\mathcal{M}_\eta\Phi(\boldsymbol{s})$ directly change the target distribution, the performance of DB strategy may be strongly deteriorated due to the misestimated action preferences function, especially in the early stage of the RL process.

To tackle this issue, we proposed the second strategy, mean of Boltzmann softmax (MB). Instead of calculating the Boltzmann softmax terms as distributions, MB strategy simply calculated the expectation of $\mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{ss}'} + \gamma\mathcal{M}_\eta\Phi_t(\boldsymbol{s}') - \mathcal{M}_\eta\Phi_t(\boldsymbol{s})$ as the shift of the given initial distribution $\Phi(\boldsymbol{s},\boldsymbol{a})$. Although it removes the distribution information in Boltzmann softmax operator, its robustness in RL framework is improved without the capriciously changed distribution in DB strategy. The evaluation of both DB and MB strategies in KLC51 was detailed in Section IV.

C. Kullback-Leibler Divergence-regularized Distributional RL

In this subsection, we detailed the workflow of the proposed KLC51 following Algorithm 1. Be similar with DQN [6], the proposed method employs the main network with parameters $\boldsymbol{\theta}$ and target network with fixed parameters $\boldsymbol{\theta}^-$ to stabilize the learning. $\boldsymbol{\theta}^-$ copies values from $\boldsymbol{\theta}$ every C steps. Before learning, a replay buffer of samples $\mathcal{D}$ with size E is initialized and warmed up by N_{initial} collected by a random policy. The training procedure has N steps. At each step t, the agent first interacts with the environment. It observes the current state $\boldsymbol{s}_t$ and determines the optimal action based on the current

Algorithm 1: KL divergence-regularized C51

Input: C, E, J, N, N_{initial}, V_{MIN}, V_{MAX}, η, α, γ
Initialize replay buffer memory $\mathcal{D}$ with size E
Initialize the weights of main networks $\boldsymbol{\theta}$
Initialize weights of target networks $\boldsymbol{\theta}^- = \boldsymbol{\theta}$
# Pre-train phase
for $k = 1$ *to* $N_{initial}$ **do**
  Randomly collect samples into replay buffer:
  $\mathcal{D} \leftarrow (\boldsymbol{s}_k, \boldsymbol{a}_k, \mathcal{R}^{\boldsymbol{a}_k}_{\boldsymbol{s}_k\boldsymbol{s}_{k+1}}, \boldsymbol{s}_{k+1})$
for $t = 1$ *to* N **do**
  if $t\%C == 0$ **then**
    set $\boldsymbol{\theta}^- = \boldsymbol{\theta}$
  # Interaction phase
  Observe current state $\boldsymbol{s}_t$
  Determine $\boldsymbol{a}_t$ by a softmax policy $\bar{\pi}_{explore}(\boldsymbol{a}_t|\boldsymbol{s}_t)$
  Execute $\boldsymbol{a}_t$, obtain $\boldsymbol{s}_{t+1}$ and $\mathcal{R}^{\boldsymbol{a}_t}_{\boldsymbol{s}_t\boldsymbol{s}_{t+1}}$
  $\mathcal{D} \leftarrow (\boldsymbol{s}_t, \boldsymbol{a}_t, \mathcal{R}^{\boldsymbol{a}_t}_{\boldsymbol{s}_t\boldsymbol{s}_{t+1}}, \boldsymbol{s}_{t+1})$
  # Training phase
  Randomly select J samples from $\mathcal{D}$
  for $j = 1$ *to* J **do**
    if *use DB strategy* **then**
      Add Boltzmann softmax distributions:
      $\hat{\Phi}'(\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\theta}^-) = \hat{\Phi}(\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\theta}^-) - \mathcal{M}_\eta\hat{\Phi}(\boldsymbol{s}_j, \cdot, \boldsymbol{\theta}^-) + \gamma\mathcal{M}_\eta\hat{\Phi}(\boldsymbol{s}_{j+1}, \cdot, \boldsymbol{\theta}^-)$
      Shift by the reward:
      $\hat{\Phi}'(\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\theta}^-) = \hat{\Phi}'(\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\theta}^-) + \mathcal{R}^{\boldsymbol{a}_j}_{\boldsymbol{s}_j\boldsymbol{s}_{j+1}}$
    if *use MB strategy* **then**
      Calculate the expected action preferences:
      $\Psi(\boldsymbol{s}_j, \cdot, \boldsymbol{\theta}^-) = \mathbb{E}[\hat{\Phi}(\boldsymbol{s}_j, \cdot, \boldsymbol{\theta}^-)]$
      $\Psi(\boldsymbol{s}_{j+1}, \cdot, \boldsymbol{\theta}^-) = \mathbb{E}[\hat{\Phi}(\boldsymbol{s}_{j+1}, \cdot, \boldsymbol{\theta}^-)]$
      Shift by the expected action preferences:
      $\hat{\Phi}'(\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\theta}^-) = \hat{\Phi}(\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\theta}^-) - \mathcal{M}_\eta\hat{\Psi}(\boldsymbol{s}_j, \cdot, \boldsymbol{\theta}^-) + \gamma\mathcal{M}_\eta\hat{\Psi}(\boldsymbol{s}_{j+1}, \cdot, \boldsymbol{\theta}^-)$
      Shift by the reward:
      $\hat{\Phi}'(\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\theta}^-) = \hat{\Phi}'(\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\theta}^-) + \mathcal{R}^{\boldsymbol{a}_j}_{\boldsymbol{s}_j\boldsymbol{s}_{j+1}}$
    Project target to pre-aligned atoms:
    $\hat{\Phi}'(\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\theta}^-) \rightarrow \hat{\Phi}'_{proj}(\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\theta}^-)$
    Calculate TD error via KL divergence:
    $L(\boldsymbol{\theta}) = D_{\text{KL}}(\hat{\Phi}'_{proj}(\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\theta}^-) \| \hat{\Phi}(\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\theta}))$
    Reduce $L(\boldsymbol{\theta})$ by updating $\boldsymbol{\theta}$ via gradient descent optimization methods with learning rate α.
return $\hat{\Phi}(\cdot, \cdot, \boldsymbol{\theta})$

knowledge following a Boltzmann softmax policy:

$$\bar{\pi}_{explore}(\boldsymbol{a}|\boldsymbol{s}) = \frac{\exp(\eta_e \mathbb{E}[\hat{\Phi}_t(\boldsymbol{s}, \boldsymbol{a})])}{\sum_{\boldsymbol{a}' \in \mathcal{A}} \exp(\eta_e \mathbb{E}[\hat{\Phi}_t(\boldsymbol{s}, \boldsymbol{a}')])} \tag{27}$$

where η_e controls the randomness of the exploration policy. The agent then operates the selected action $\boldsymbol{a}_t$ and receives the next step state $\boldsymbol{s}_{t+1}$ and reward signal $\mathcal{R}^{\boldsymbol{a}_t}_{\boldsymbol{s}_t\boldsymbol{s}_{t+1}}$. The whole transition $(\boldsymbol{s}_t, \boldsymbol{a}_t, \mathcal{R}^{\boldsymbol{a}_t}_{\boldsymbol{s}_t\boldsymbol{s}_{t+1}}, \boldsymbol{s}_{t+1})$ is stored in $\mathcal{D}$.

After interaction, the agent moves to the training phase. Following the description in Section III-B, the DB strategy will calculate the distributional Boltzmann softmax terms $-\mathcal{M}_\eta\hat{\Phi}(\boldsymbol{s}_j, \cdot, \boldsymbol{\theta}^-)$ and $\gamma\mathcal{M}_\eta\hat{\Phi}(\boldsymbol{s}_{j+1}, \cdot, \boldsymbol{\theta}^-)$. After projecting to the pre-aligned, these distributions are added to the target action preferences function $\hat{\Phi}'(\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\theta}^-)$. It is then shifted by the reward $\mathcal{R}^{\boldsymbol{a}_t}_{\boldsymbol{s}_t\boldsymbol{s}_{t+1}}$. Using the simplified MB strategy, the expectation of the Boltzmann softmax terms, $-\mathcal{M}_\eta\hat{\Psi}(\boldsymbol{s}_j, \cdot, \boldsymbol{\theta}^-)$ and $\gamma\mathcal{M}_\eta\hat{\Psi}(\boldsymbol{s}_{j+1}, \cdot, \boldsymbol{\theta}^-)$ are first calculated. The target action preferences function $\hat{\Phi}'(\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\theta}^-)$ is shifted by these expected terms and the reward $\mathcal{R}^{\boldsymbol{a}_t}_{\boldsymbol{s}_t\boldsymbol{s}_{t+1}}$. After obtaining $\hat{\Phi}'(\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\theta}^-)$, KLC51 projects it to the pre-aligned atoms $\hat{\Phi}'_{proj}(\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\theta}^-)$. The TD error $L(\boldsymbol{\theta})$ is calculated as the KL divergence between distributions $\hat{\Phi}'_{proj}(\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\theta}^-)$ and $\hat{\Phi}(\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\theta})$. It is reduced by any gradient descent optimization methods with learning parameters α following the settings of related works [6], [11].

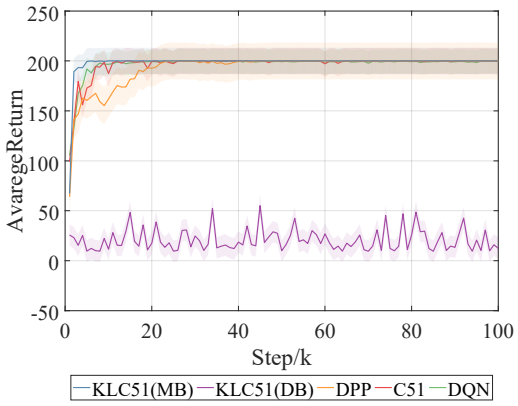

Fig. 4. Learning curves of the proposed KLC51 and other baseline approaches in CartPole-v0 task.

IV. Experiments

A. Experimental Setting

In this section, we evaluated the proposed KLC51 in three benchmark tasks in OpenAI gym [1] with increasing complexity, cartpole-v0, Lunarlander-v2 and Fetchreach-v1. The proposed KLC51 was developed by Tensorflow [26] and Tensorflow Agents [27]. DQN [6], C51 [11] and DPP [17] were selected as the baseline approaches. We used the implementations of DQN and C51 in Tensorflow Agents, and implemented DPP with neural networks in the same framework following [21]. All experiments were conducted on a computational server with Xeon-W2275 CPU, 64 GB memory and Ubuntu 20.04 OS. The common parameters of all RL approaches were set as $N_{\text{initial}} = 1000$, $N = 10^5$, $\gamma = 0.99$. Five independent trials were conducted with different random seeds to collect statistical evidence.

B. Evaluation of the Distributional TD Error in KLC51

We started from a simple control task, CartPole-v0 which has 2-dimensional state and 2 discrete actions to evaluate the Distributional TD Error with different strategies. The proposed method and other baseline approaches shared a

[1] https://gym.openai.com/

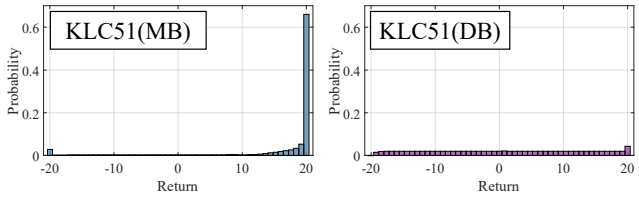

Fig. 5. Action preferences function distribution of one example state-action pair learned in KLC51 using MB and DB strategies in CartPole-v0 task.

same network structure, one fully-connected layer with 100 neurons and ReLU activation function. The learning rate was set as $\alpha = 10^{-3}$, the parameters of replay buffer was set as $J = 64, E = 10^5$, the main and target networks shared weights every $C = 5$ steps. KLC51 and DPP utilized Boltzmann exploration policy with $\eta_e = 0.1$ while DQN and C51 employed a $\epsilon-$greedy exploration policy with $\epsilon = 0.1$. We set the number of atoms as 51 and chose $V_{MAX} = -V_{MIN} = 20$ for both KLC51 and C51. The KL divergence penalty term η in KLC51 and DPP was set to 0.1.

The learning curves of average return using the evaluation policy over all approaches were demonstrated in Fig. 4. C51 and DQN had close performances which required more than 10k steps to converge and slightly fluctuated during the learning process as they shared a very similar network structure. On the other hand, DPP achieved a slower convergency (more than 20k steps) and could not reach our expectation of the regularization of KL divergence. One possible reason is the original DPP [17] was not designed for the neural network structure, the replay buffer and main/target networks introduced additional randomness which turned to over-cautious policy update with low data-efficiency. As comparison, KLC51 using MB strategy significantly outperformed other baseline approaches in not only convergency speed (less than 5k steps), but also the robustness during training (no fluctuation in average return). Processing the additional randomness in a distributional perspective, it successfully improved the data-efficiency under the neural network structure. On the other hand, the DB strategy could not work in KLC51 and resulted in a fluctuating learning curve without convergency. This result is consistent to our discussion in Section III-B that directly changing the target distribution by the distributional Boltzmann softmax terms may deteriorate the learning performance, especially with misestimated value distribution in the early stage of RL. As one case study, an example state-action pair's action preferences function distributions of both DB and MB strategies were compared in Fig. 5. It is clearly observed that the distribution learned in MB strategy successfully captured a distribution around high returns, while the one learned by DB strategy was close to the uniform distribution with less information.

C. Evaluation of the Learning Capability

In the second experiment, the learning capability and the effect of KL divergence regularization were evaluated in a more complex control problem, LunarLander-v2 which has 8-dimensional state and 4 discrete actions. The proposed method

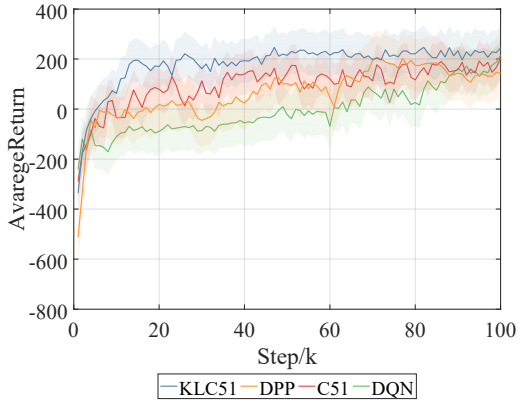

Fig. 6. Learning curves of the proposed KLC51 and other baseline approaches in LunarLander-v2 task.

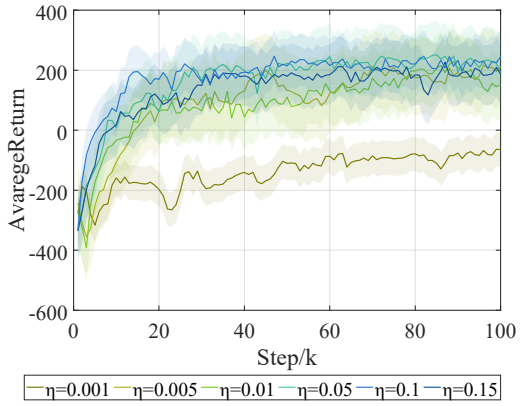

Fig. 7. Learning curves of the proposed KLC51 with different parameter η in LunarLander-v2 task.

and other baseline approaches shared a same network structure with two fully-connected layers with $[256, 256]$ neurons and ReLU activation function. The learning rate was set as $\alpha = 10^{-4}$, the parameters of replay buffer was set as $J = 256, E = 10^4$, the main and target networks shared weights every $C = 100$ steps. KLC51 and DPP utilized Boltzmann exploration policy with $\eta_e = 0.05$ while DQN and C51 employed a $\epsilon-$greedy exploration policy with $\epsilon = 0.1$. We set the number of atoms as 51 and chose $V_{MAX} = -V_{MIN} = 200$ for both KLC51 and C51. The KL divergence penalty term η in KLC51 and DPP was set to 0.1.

The learning curves of the average returns using the evaluation policy were compared in Fig. 6. DQN had the worst performance with neither the distributional value fucntion nor the KL divergence regularization. Compared with DPP and C51 that converged to relatively low returns after 80k steps, the proposed KLC51 using MB strategy successfully improved over 75% data-efficiency with better robustness: it quickly converged to the highest average return within 20K steps and maintained stability after convergency. This result demonstrated the advantages of proposed KLC51 in both data-efficiency and robustness in a complex control task by employing the KL divergence regularization in a distributional

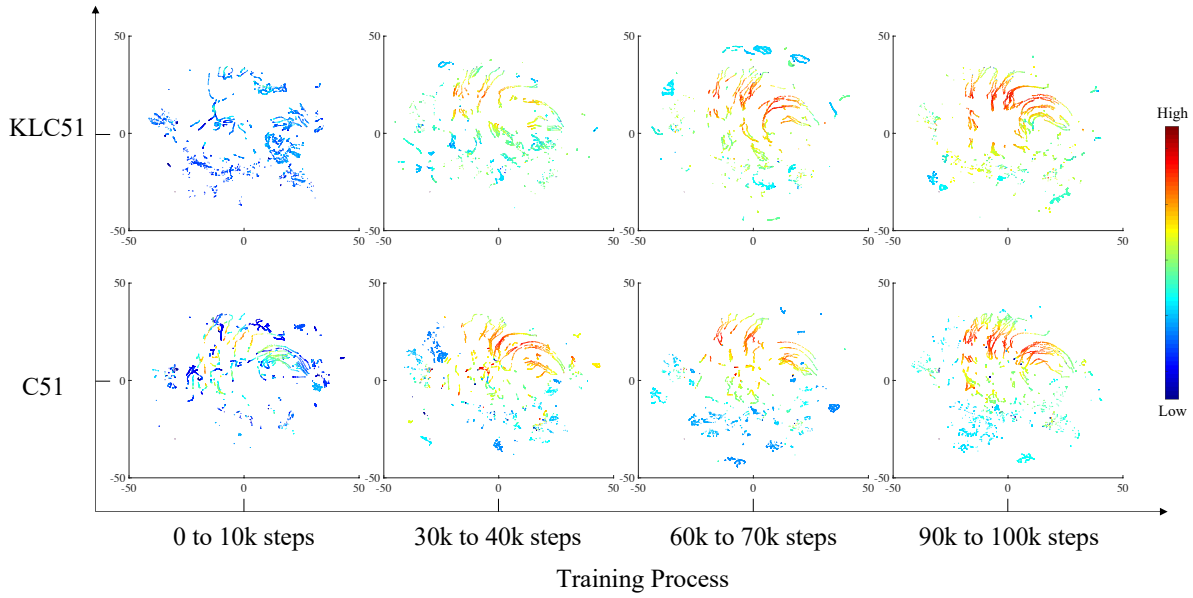

Fig. 8. Samples colloected by KLC51 and C51 during the learning procedure. The high-dimensional samples were embedded by t-SNE, the color presents the expectation of value function.

perspective.

The effect of parameter η which controls the weight of KL divergence regularization in KLC51 was studied in this subsection. The learning curves of average returns with different setting of η were shown in Fig. 7. This result is similar to the one of DPP-based engineering implementations [20], [21]. A small η not only contributed to a strong penalty items to the over-large policy update but also slowed down the convergence. With an increasing η from 0.001 to 0.1, the convergency speed and the final average return were all improved. On the other hand, an over-large $\eta = 0.15$ in KLC51 could damage data-efficiency and learning capability and turned to a close performance to C51 as the over-large policy is not sufficiently constrainted. This result indicated the importance of properly selecting the parameter η in KLC51's engineering application.

We finally explored the effect of KL divergence regularization to the sampling behaviors in KLC51. The seperated 10k high-dimensional samples collected within $(0, 10]$, $(30, 40]$, $(60, 70]$, $(90, 100]$ thousand steps were illustrated using a reliable dimensionality reduction technique, t-distributed stochastic neighbor embedding (t-SNE) [28] in Fig. 8 where the colors from blue to red presented expected value function with low to high values. At the early stage of learning (10k step), C51 explored more samples with high expected value (yellow and red dots) while KLC51 was restricted to low value samples by the KL divergence penalty term. From 40k step to 100k step, KLC51 smoothly imporved its policy to collect more samples with high values while avoiding the one with extremely low value. On the other hand, the larger variance of expected values in C51's samples indicated an insufficient exploration caused by the drastically changed policy without constraint which resulted in poor data-efficiency and stability compared with KLC51.

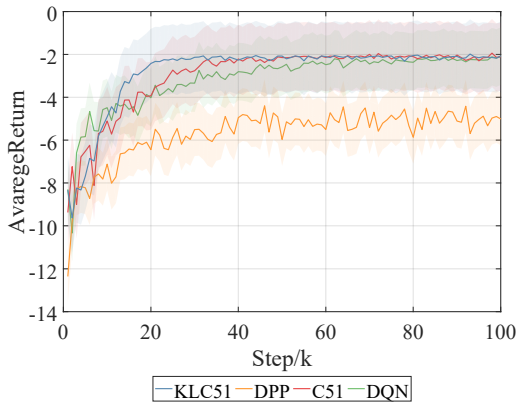

Fig. 9. Learning curves of the proposed KLC51 and other baseline approaches in Fetchreach-v1 task.

D. Evaluation of the KL-regularized Behaviors

The last experiment focused on the FetchReach-v1 task with a 13-dimensional dictionary-based observation space and a 4-dimensional continuous action space in order to indicate the control performances of KLC51 in robot control domain. The continuous action space was discretized to 7 discrete actions. The proposed method and other baseline approaches shared a same network structure with three fully-connected layers with $[256, 256, 256]$ neurons and ReLU activation function. The learning rate was set as $\alpha = 10^{-4}$, the parameters of replay buffer was set as $J = 64, E = 10^5$, the main and target networks shared weights every $C = 5$ steps. KLC51 and DPP utilized Boltzmann exploration policy with $\eta_e = 0.2$ while DQN and C51 employed a $\epsilon-$greedy exploration policy with $\epsilon = 0.1$. We set the number of atoms as 51 and chose $V_{MAX} = -V_{MIN} = 10$ for both KLC51 and C51. The KL

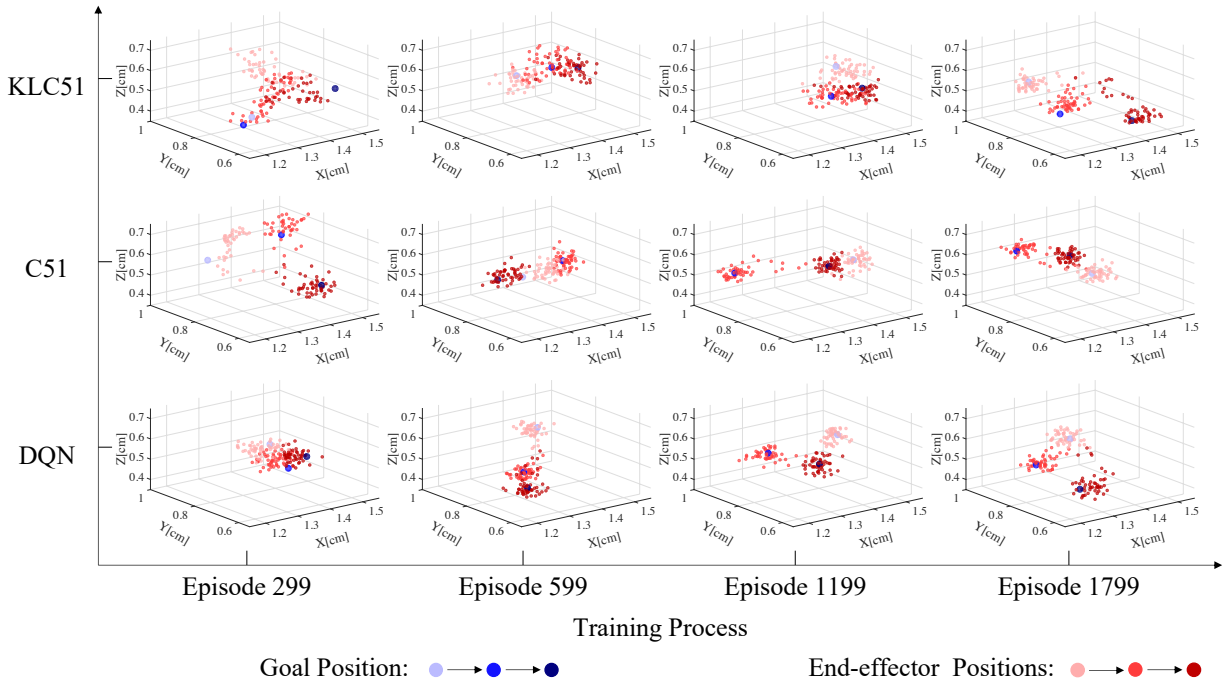

Fig. 10. End-effector positions (red dots) and the goal positions (blue dots) in continuous three steps (from light to dark colors) sampled in different learning procedure using KLC51, C51 and DQN.

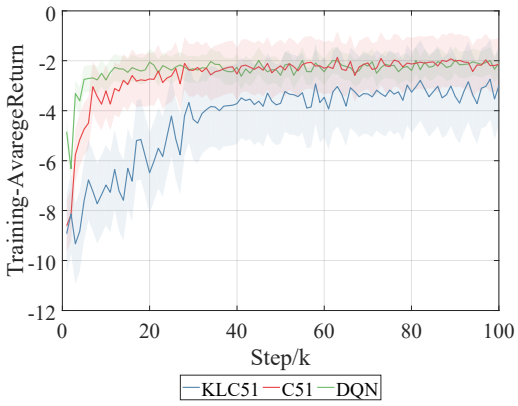

Fig. 11. Exploration policies' average returns during training using the proposed KLC51 and other baseline approaches in Fetchreach-v1 task.

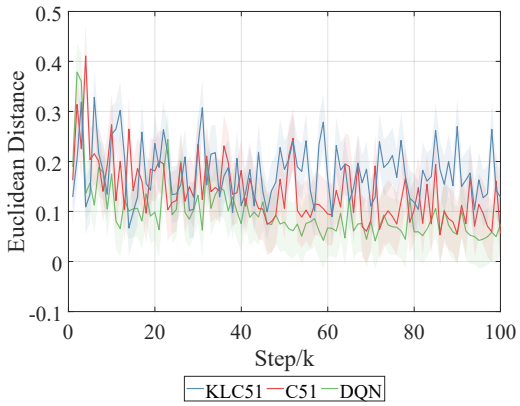

Fig. 12. Average Euclidean distance between the end-effector and target position during training using the proposed KLC51 and other baseline approaches in Fetchreach-v1 task.

divergence penalty term η in KLC51 and DPP was set to 0.15.

According to the learning curves of average return using the evaluation policy compared in Fig. 9, although KLC51 could not outperformed C51 and DQN in the first 10k step, it quickly converged to the highest return -2 within about 20k steps and achieved 50% and 66.7% better data-efficiency compared with C51 and DQN which converged within 40k and 60k steps respectively. DPP had the worst performance, it slowly converged to a lower average return within 40k with a fluctuated learning curve.

In the next step, we studied the exploration behaviors in a more intuitive way to investigate the effect of KL divergence regularization in a distributional perspective. Define 50 steps as one episode, the whole training process of 100k steps had 2000 episodes. The end-effector positions (red dots) and the corresponding goal position (blue dots) in three continuous episodes $e-1$, e and $e+1$ (from light to dark colors) were demonstrated in Fig. 10 with four different phases of training $e = [299, 599, 1199, 1799]$. It is clearly observed that C51 and DQN rapidly switched their sampling distributions to track the goal positions from episode $e-1$ step to $e+1$. As the algorithm converged, the sampling policy focused on a shrinking local space around the goal positions. As comparison, a more extensive exploration was encouraged in KLC51. Regularized by the KL divergence between the current and previous policies, KLC51 naturally focused on a larger area between each goal positions to efficiently collect samples over the task

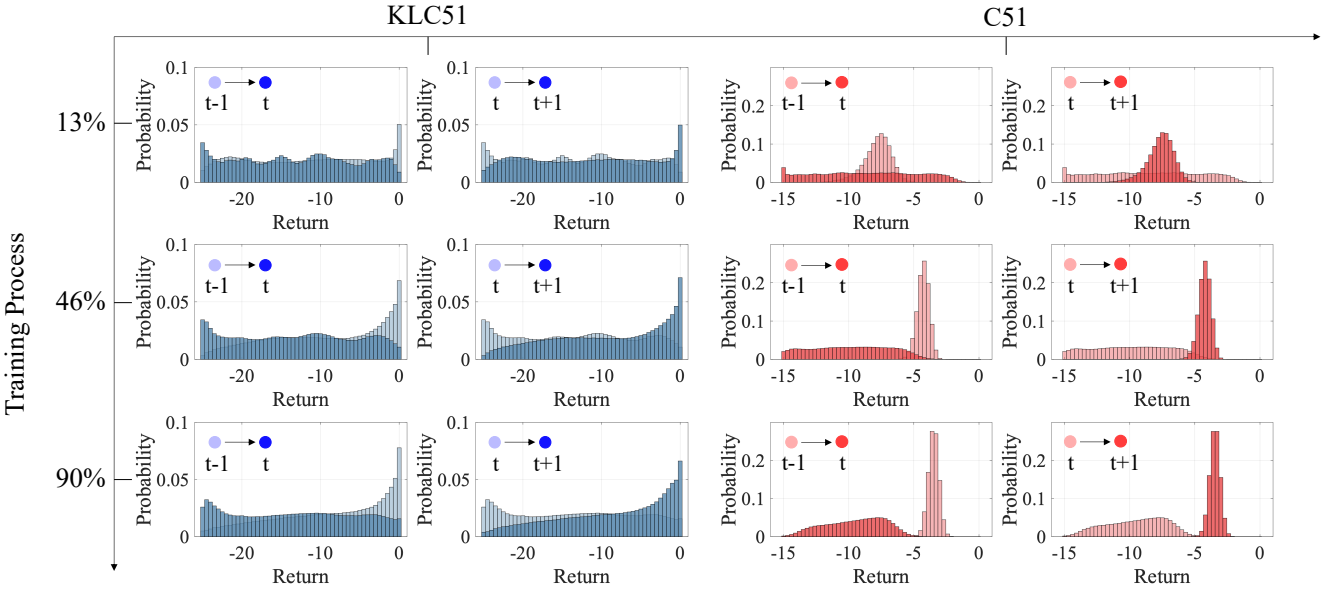

Fig. 13. Update of distributional value functions in KLC51 and C51 over three continuous update steps where the main and target networks shared weights in different learning procedure. Light and dark colors represent the distributions in current and next steps.

space. On the other hand, directly modeling the expected value function with such an extensive exploration resulted in both slow convergency and poor control performance in DPP due to the randomness from the wide range of sampling, the replay buffer and the structure of main and target neural networks while KLC51 successfully achieved great data-efficiency and robustness by handling these randomness in a distributional perspective following C51. The average rewards and the Euclidean distances between end-effector and the goal shown in Figs. 11 and 12 supported the results in Figs. 10 and 9: KLC51's extensive exploration with lower average reward contributed to better learning performance using the evaluation policy compared with the baseline approaches.

Finally we investigated the effect of KL divergence regularization in the update of distributional value function. Both Φ and Z distributions in KLC51 and C51 of one example state-action pair over three continuous update steps when the main and target networks shared the weights were demonstrated in Fig. 13 as a case study to indicate the robustness against neural networks' stochasticity given by the KL divergence regularization. The distributions in current and next steps were indicated by light and dark colors. In the horizontal view of Fig. 13, it is clearly observed in C51 that the weights copy in the target network every C steps turned to strongly flattened and shifted distributional value functions between steps $t-1$ and $t+1$. These different distributions therefore introduced additional uncertainties to the learning process. As comparison, the over-large update of distributional value functions was alleviated in KLC51 through the Boltzmann softmax operator. The distributions updated in step t shared more information of the previous step than C51 in a relatively flat shape which not only contributed to an exclusive exploration, but also captured the stochasticity caused by the nature of neural networks. In the vertical view of Fig. 13, the update of distributional value function from 13% to 90% learning procedure also enjoyed better smoothness with the KL divergence regularization. Compared with C51 that kept shifting and shrinking its distribution during the learning, KLC51 quickly converged to a high average return according to Fig. 9 while maintaining a flatten distribution without over-large update. This result indicated the smooth update and stable learning reported by [20], [22] were successfully inherited by the proposed approach in a distributional perspective.

V. Conclusions

In this article, a novel RL approach, KLC51 was proposed to tackle the issues of stability and data-efficiency in RL by integrating the advantages of both distributional RL approach C51 and the KL divergence-regularized RL approach DPP. The proposed method derived the Bellman equation and TD errors regularized by KL divergence in a distribution perspective and explored the approximated update strategies to map the corresponding Boltzmann softmax term in DPP to distributions. With a distributional value function, KLC51 successfully stabilized and accelerated the learning procedure by properly handling the stochasticity from not only the exclusive exploration regularized by KL divergence but also its neural networks structure. Evaluated by several benchmark tasks with different complexity, KLC51 demonstrated its significant superiority in both learning stability and data-efficiency compared with related baseline approaches. We further studied and illustrated the positive effect of the KL divergence regularization in distributional RL including the exploration behaviors and the update of distributional value function. All these results indicated the potential of KLC51 as an emerging stable and data-efficient RL approach for various control problems.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[2] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[3] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal and autonomous control using reinforcement learning: A survey," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 6, pp. 2042–2062, 2017.

[4] W. Zhu, X. Guo, D. Owaki, K. Kutsuzawa, and M. Hayashibe, "A survey of sim-to-real transfer techniques applied to reinforcement learning for bioinspired robots," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[5] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[7] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[8] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*, pp. 1587–1596, PMLR, 2018.

[9] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 3389–3396, IEEE, 2017.

[10] A. A. Rusu, M. Večerík, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets," in *Conference on Robot Learning*, pp. 262–270, PMLR, 2017.

[11] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *International Conference on Machine Learning*, pp. 449–458, PMLR, 2017.

[12] P. Zhang, X. Chen, L. Zhao, W. Xiong, T. Qin, and T.-Y. Liu, "Distributional reinforcement learning for multi-dimensional reward functions," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[13] J. Xie and W. Sun, "Distributional deep reinforcement learning-based emegercy frequency control," *IEEE Transactions on Power Systems*, pp. 1–1, 2021.

[14] K. Min, H. Kim, and K. Huh, "Deep distributional reinforcement learning based high-level driving policy determination," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 3, pp. 416–424, 2019.

[15] C. Lyle, M. G. Bellemare, and P. S. Castro, "A comparative analysis of expected and distributional reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 4504–4511, 2019.

[16] M. Rowland, M. Bellemare, W. Dabney, R. Munos, and Y. W. Teh, "An analysis of categorical distributional reinforcement learning," in *International Conference on Artificial Intelligence and Statistics*, pp. 29–37, PMLR, 2018.

[17] M. G. Azar, V. Gómez, and H. J. Kappen, "Dynamic policy programming," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 3207–3245, 2012.

[18] N. Vieillard, T. Kozuno, B. Scherrer, O. Pietquin, R. Munos, and M. Geist, "Leverage the average: an analysis of kl regularization in reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12163–12174, 2020.

[19] T. Kozuno, E. Uchibe, and K. Doya, "Theoretical analysis of efficiency and robustness of softmax and gap-increasing operators in reinforcement learning," in *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2995–3003, PMLR, 2019.

[20] Y. Cui, T. Matsubara, and K. Sugimoto, "Kernel dynamic policy programming: Applicable reinforcement learning to robot systems with high dimensional states," *Neural networks*, vol. 94, pp. 13–23, 2017.

[21] Y. Tsurumine, Y. Cui, E. Uchibe, and T. Matsubara, "Deep reinforcement learning with smooth policy update: Application to robotic cloth manipulation," *Robotics and Autonomous Systems*, vol. 112, pp. 72–83, 2019.

[22] L. Zhu, Y. Cui, G. Takami, H. Kanokogi, and T. Matsubara, "Scalable reinforcement learning for plant-wide control of vinyl acetate monomer process," *Control Engineering Practice*, vol. 97, p. 104331, 2020.

[23] L. Zhu, G. Takami, M. Kawahara, H. Kanokogi, and T. Matsubara, "Alleviating parameter-tuning burden in reinforcement learning for large-scale process control," *Computers & Chemical Engineering*, p. 107658, 2022.

[24] A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, *et al.*, "Conditional image generation with pixelcnn decoders," *Advances in neural information processing systems*, vol. 29, 2016.

[25] E. Todorov, "Linearly-solvable markov decision problems," in *Advances in neural information processing systems (NIPS)*, pp. 1369–1376, 2006.

[26] M. Abadi *et al.*, "TensorFlow: A system for Large-Scale machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pp. 265–283, 2016.

[27] S. Guadarrama *et al.*, "TF-Agents: A library for reinforcement learning in tensorflow." https://github.com/tensorflow/agents, 2018. [Online; accessed 25-June-2019].

[28] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.