Latent Context Based Soft Actor-Critic

Yuan Pu, Shaochen Wang, Xin Yao, Bin Li
University of Science and Technology of China, Hefei, China
Email: {puyuan, samwang, xinyao}@mail.ustc.edu.cn, binli@ustc.edu.cn

***Abstract*—The performance of deep reinforcement learning methods prone to degenerate when applied to tasks requiring relatively longer horizon memory or with highly variable dynamics. In this paper, we utilize the probabilistic latent context variables motivated by recent Meta-RL materials, and propose the Latent Context based Soft Actor-Critic (LC-SAC) approach to address aforementioned issues. The latent context is capable to encode information about both the agent's previous behaviors and the dynamics of the current undergoing environment, which empirically believed to be beneficial for efficient policy optimization. Experiment results demonstrate that LC-SAC can achieve comparable performance with SAC on a collection of continuous control benchmarks and outperforms SAC in some particular tasks with above two characteristics. Moreover, we also introduce a simple but general procedure to integrate LC-SAC with diverse-quality demonstrations to enable efficient reuse of human prior knowledge, and finally achieve competitive performance with comparatively small number of interactions with environments.**

I. Introduction

Recent years, deep reinforcement learning (DRL) algorithms [1] have present many impressive results in many challenging domains, such as Atari 2600 arcade games [2], board games [3], robot manipulation tasks [4] [5] .

To achieve sample efficiency and robust performance, many works have been advanced. Especially, soft actor-critic algorithm [6] achieves state-of-the-art performance on many challenging continuous control benchmarks, which is an off policy actor-critic setting RL algorithm based on the maximum entropy reinforcement learning framework. The actor reuses past experience through off-policy optimization and acts maximize expected reward while also maximizing entropy to increase exploration.

However, when one task implicitly requires long-horizon sequential decisions or varies with initial conditions or specific dynamics, the conventional algorithms usually could not solve it because the agent lacks effective memory mechanism and cannot obtain the necessary long-horizon information. For example, in Mujoco task [22], Striker-v2, the simulated mechanical arm aims to strike a ball into a target locket. The initial location of the target locket changes randomly every episode. Keeping the location of this target locket in mind and planning a move trajectory is crucial to successfully accomplish this mission. A natural improvement could be explicitly increase the agent's memory length. In previous research [7], their proposed Deep Recurrent Q-Network (DRQN), successfully integrates information through time and replicates DQN's performance on discrete Atari domains. However through empirical experiments, they found that using recurrent neural networks can not guarantee systematic performance improvement.

Recently, in Meta-RL domain, [8] [9] [10] have proposed to encode task's salient identification information to a latent embedding space to reuse experience among different similar tasks and achieve both meta training and adaptation efficiency.

Motivated by these insights, in this paper, we incorporate probabilistic latent context variables into SAC to tackle the aforementioned issues, and propose the Latent Context based Soft Actor-Critic (LC-SAC) approach. The performance of the LC-SAC is evaluated on representative single continuous control tasks. And the capability of latent context techniques to solve tasks that require longer-horizon memory is also investigated.

Experiment results show that LC-SAC can achieve comparable performance with SAC in many continuous tasks and can outperform SAC in certain particular tasks, which we believe that is because the latent context goes beyond providing the agent with relatively long-horizon memory, and potentially enabling better representation learning in the anterior part of the value and policy neural networks.

In addition, conventional RL algorithms usually require very large number of interactions with environments to achieve acceptable performance. Sometimes, these interactions are limited and economically expensive. [11] [12] show that human's prior knowledge such as demonstrations can be used to accelerate agents' learning, which can be implemented through behavior cloning methods to simply mimic human experts' behavior or inverse reinforcement learning [13] [14] to learn a reward function that contains the task's characteristics. However, these methods typically require the expert demonstrations, which are usually hard to get and may be sub-optimal. In present paper, we show that simply combining LC-SAC with different-quality demonstrations can achieve comparable results in fewer interactions with environments and ultimately outperform the best performance of demonstrations in most tasks.

The main contributions in this paper are:

- proposing the LC-SAC method to explicitly memorize recently acquired knowledge through learning a latent context variable and exploring its practical impacts on the original SAC on a variety of continuous Mujoco benchmarks.
- combining LC-SAC with diverse-quality demonstrations to effectively learn from both interactions with environments and human knowledge, and competitive performance is observed.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

Our code is based on this open-source implementation [1].

II. RELATED WORKS

A. Soft Actor-Critic

Before introducing SAC, we first present the deep reinforcement learning problem definition. Reinforcement learning problem is often formulated as a *Markov Decision Process* (MDP), $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma)$. When the RL agent interacting with the environment, at each step, the agent observes a state $\mathbf{s}_t \in \mathcal{S}$, where $\mathcal{S}$ is the state space, and chooses an action $\mathbf{a}_t \in \mathcal{A}$, according to policy $\pi(\mathbf{a}_t|\mathbf{s}_t)$, where $\mathcal{A}$ is the state space, then the agent receives a reward $r(\mathbf{s}_t, \mathbf{a}_t)$ and the environment transforms to next state $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$. Note that the environment's dynamics is contained in both the transition probabilities p and reward functions r, which are usually unknown to the agent.

The objective of maximum entropy reinforcement learning framework [15] is to maximize the discounted expected total reward plus the expected entropy of the policy:

$$J(\pi) = \sum_{t=0}^{T} \mathbf{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} \left[r(\mathbf{s}_t, \mathbf{a}_t) + \alpha \mathcal{H}(\pi(\cdot|\mathbf{s}_t)) \right] \quad (1)$$

, where γ is the discounted factor, $\rho_\pi(\mathbf{s}_t, \mathbf{a}_t)$ denotes the state-action marginal of the trajectory distribution induced by the policy $\pi(\mathbf{a}_t|\mathbf{s}_t)$.

Soft Actor-Critic (SAC) [6] is an off-policy actor-critic method in the maximum entropy reinforcement learning framework. It utilizes an actor-critic architecture with separate policy and value networks, an off-policy paradigm that enables reuse of previously collected data, and entropy maximization to enable effective exploration. In contrast to other off-policy algorithms, SAC is quite stable and achieving state-of-the-art results on a range of continuous control benchmarks. To verify our hypothesis that latent context could increase the the agent's memory length and implicitly benefit the representation learning, our modification is based on SAC.

B. RL with Recurrent Neural Networks

One way to tackle memory-required problems is to use recurrent neural networks. Many prior work [7] [16] [17] have combined LSTM networks with policy gradient methods to solve POMDPs. [7] investigated the effects of adding recurrence to a Deep Q-Network (DQN), successfully integrates information through time and receives comparable scores with DQN on most standard Atari games. Meanwhile, [16] proposed to extend deterministic policy gradient and stochastic value gradient to recurrent neural networks variants to solve challenging memory problems such as the Morris water maze.

Recently, [17] empirically investigated the training of RNN-based RL agents from distributed prioritized experience replay, and obtained remarkably good results in Atari games domain. But it also showed representational drift and recurrent state staleness problem is exacerbated in the continuous training setting, and ultimately results in diminished training stability and performance. Meanwhile in [8], experimentally they found that straightforward incorporation of recurrent policies with off-policy learning is difficult, especially in continuous settings.

C. Probabilistic Meta-RL

In Meta-RL materials, many approaches [18] [19] [20] have been proposed to transfer among different similar tasks. MAESN [19] combines structured stochasticity with MAML [18] by learning exploration strategies from prior experience, resulting in fast adaptation to new tasks.

The most related work with us is PEARL [8] algorithm. They represent task contexts with probabilistic latent variables, which encode the commonness of different tasks in some meaning. This probabilistic interpretation enables temporally extended exploration behaviors that enhance adaptation efficiency while requiring far less experience.

Our proposed LC-SAC algorithm is a variant based on PEARL. The main difference is that, PEARL aims to solve Meta-RL problems, so the latent context variables $\mathbf{c}$ encode salient identification information about the task, while in our LC-SAC, the latent context is trained to memorize the recent information about the dynamics of the current undergoing environment and the agent's past actions (or behaviors).

Our first main concern in this paper is if adding latent context will impair conventional RL algorithms' performance in simple continuous tasks in different experimental parameters settings and if latent context can be used to improve original SAC algorithms' performance in tasks that require long-horizon memory for decision.

D. Learning from Human Demonstrations

Reinforcement learning typically require numerous interactions with environments before obtaining acceptable performance. However, in some conditions, for example, auto-driving, the cost of interactions with environments is huge even unacceptable. In these cases, prior work [11] [12] [13] [14] [21] try to leverage existing human knowledge (usually demonstrations) to reduce the strong dependency on interactions with environments.

However, demonstrations may be sub-optimal and not cover all states space. So simply mimic the demonstrations through behavior cloning is not consistent to obtain a comparable results in most cases. Another way is inverse reinforcement learning methods, which try to infer a intrinsic reward function from different demonstrations and then use it to optimize policy based on conventional forward RL, which is a viable method in sparse reward settings but usually need much more computational cost.

So our second main concern in this paper is how to integrate the usage of the prior knowledge and policy optimization from interactions with environments. Empirically, we find that our LC-SAC can be simply but effectively combined with diver-quality demonstrations and achieve comparable results with SAC in most continuous control benchmarks.

[1] https://github.com/katerakelly/oyster

III. METHODS

In this section, through introducing latent context variable $\mathbf{c}$ to capture information about recent knowledge of the environment's dynamics and the agent's past behaviors, we first present our new proposed LC-SAC algorithm. In addition, [8] indicated that, in an off-policy meta-RL method with the probabilistic context, the data used to train the latent context encoder need not be the same as the data used to train the policy. Motivated by this hypothesis, next we presented LC-SAC-DD procedure, which can effectively integrate the LC-SAC algorithm with diverse-quality demonstrations to largely reduce the numbers of interacting with environments before getting achievable performance.

A. Latent Context Based Soft Actor-Critic

Notation Recall that both the transition and reward functions are usually unknown, and can be sampled by taking actions in the environment. We denote the agent's transition collected at t time step as experience $\mathbf{e}_t = (\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}'_t)$ and $\mathbf{e}_{1:H}$ as the experience collected so far, where H denotes the length of memory: *Horizon*. And we propose an latent context encoder network $q_\omega(\mathbf{c}|\mathbf{e})$, which maps the experience $\mathbf{e} \in \mathcal{E}$ to $\mathbf{c} \in \mathcal{C}$, where $\mathcal{E}$ denotes the experience space, $\mathcal{C}$ denotes the latent context space, and ω denotes latent context encoder network parameters. Note, for simplicity, in the subsequent content, we use $\mathbf{e}$ represents the experience collected in recent H steps: $\mathbf{e}_{1:H}$.

Based on original soft actor-critic algorithm, our Latent Context based Soft Actor-Critic (LC-SAC) methods utilize one state value network $V_\psi(\mathbf{s}_t, \mathbf{c})$, two soft Q-value networks $Q_{\theta_i}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{c})$, for $i \in \{1, 2\}$, a policy network $\pi_\phi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{c})$. All networks argument their input with the latent context variable $\mathbf{c}$, which encodes recent N steps experience, consisted of the information of environmental dynamics and agent's past behaviors, explicitly enabling longer-horizon memory. We also adopted the soft policy iteration framework, alternating between optimizing these networks with stochastic gradient descent.

To optimize the latent context encoder network $q_\omega(\mathbf{c}|\mathbf{e})$. We use the amortized variational inference approach same as [8]. The optimization objective is as follows:

$$J_{\mathbf{q}}(\omega) = \mathop{\mathbb{E}}_{\mathbf{c}\sim q_\omega(\mathbf{c}|\mathbf{e})} \left[\mathcal{L}_{critic} + \beta D_{\mathrm{KL}}\left(q_\omega(\mathbf{c}|\mathbf{e}) \,\|p(\mathbf{c})\right)\right] \quad (2)$$

in equation (2), the first term is conventional SAC critic loss, in our setting, specifically, $\mathcal{L}_{critic} = J_Q(\theta_1) + J_Q(\theta_2)$,

$$J_Q(\theta_i) = \mathop{\mathbb{E}}_{\substack{(\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D} \\ \mathbf{c} \sim q_\omega(\mathbf{c}|\mathbf{e})}} \left[\frac{1}{2}\left(Q_{\theta_i}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{c}) - \hat{Q}_{\theta_i}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{c})\right)^2\right] \quad (3)$$

where,

$$\hat{Q}_{\theta_i}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{c}) = r(\mathbf{s}_t, \mathbf{a}_t) + \bar{V}_\psi(\mathbf{s}'_\mathbf{t}, \mathbf{c}) \quad (4)$$

where $\mathcal{D}$ is a replay buffer containing previously sampled states and actions, and $\bar{V}_\psi$ is the soft value target network, which is trained to minimize the squared residual error:

Algorithm 1 LC-SAC

Initialize network parameters: $\omega, \theta_{1,2}, \psi, \bar{\psi}, \phi$, and two replay buffers: $\mathcal{D}_c$ for training latent context encoder, $\mathcal{D}_{rl}$ for training policy and value networks.

while not converged **do**
  for each collecting data step **do**
    clear context buffer $\mathcal{D}_c$
    for $t = 0 : N_c$ **do**
      sample $\mathbf{c} \sim p(\mathbf{c})$, rollout policy $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{c})$ to get experience $\mathbf{e}^\mathbf{c}_\mathbf{t}$
    end for
    add $\mathbf{e}^c_{t=1\ldots N_c}$ to buffer $\mathcal{D}_c$ and $\mathcal{D}_{rl}$
    for $t = 0 : N_{rl}$ **do**
      sample $\mathbf{c} \sim q_\omega(\mathbf{c}|\mathbf{e})$, rollout policy $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{c})$ to get experience $\mathbf{e_t}$
    end for
    add $\mathbf{e}_{t=1\ldots N_{rl}}$ to buffer $\mathcal{D}_{rl}$
  end for
  for each training step **do**
    sample $\mathbf{B}_c$ context batch randomly from $\mathcal{D}_c$
    sample $\mathbf{B}_{rl}$ RL batch randomly from $\mathcal{D}_{rl}$
    sample $\mathbf{c} \sim q_\omega(\mathbf{c}|\mathbf{e})$
    $\omega \leftarrow \phi - \alpha_q \nabla_\omega J_q(\omega)$
    $\theta_i \leftarrow \theta_i - \alpha_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$, for $i \in \{1, 2\}$
    $\psi \leftarrow \psi - \alpha_V \hat{\nabla}_\psi J_V(\psi)$
    $\phi \leftarrow \phi - \alpha_\pi \hat{\nabla}_\phi J_\pi(\phi)$
    $\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi}$
  end for
end while
return $q_\omega, Q_\theta, V_\psi, \pi_\phi$

$$J_V(\psi) = \mathop{\mathbb{E}}_{\mathbf{s}_t\sim\mathcal{D}} \left[\frac{1}{2}\left(V_\psi(\mathbf{s}_t, \mathbf{c}) - \hat{V}_\psi(\mathbf{s}_t, \mathbf{c})\right)^2\right] \quad (5)$$

where,

$$\hat{V}_\psi(\mathbf{s}_t, \mathbf{c}) = \mathop{\mathbb{E}}_{\mathbf{a}_t\sim\pi_\phi} \left[Q_\theta(\mathbf{s}_t, \mathbf{a}_t, \mathbf{c}) - \log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{c})\right] \quad (6)$$

Recent work theoretically proofed that the the soft (Boltzmann) policy iteration is guaranteed to improve and can converge to the optimal policy. Derived from the soft policy iteration procedure, the objective for policy update is below:

$$J_\pi(\phi) = \mathop{\mathbb{E}}_{\mathbf{s}_t\sim\mathcal{D}, \mathbf{c}\sim q_\omega(\mathbf{c}|\mathbf{e})} \left[\log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{c}) - Q_\theta(\mathbf{s}_t, \mathbf{a}_t, \mathbf{c})\right] \quad (7)$$

in equation (2)'s right hand, the second term, $p(\mathbf{c})$ is a zero mean, unit variance Gaussian probabilistic distribution. The KL divergence term is a kind of regularization aiming to enable our latent encoder network's output similar as a Gaussian distribution as much as possible when minimizing the critic loss, and implicitly increases the exploration in the latent context space and avoids the premature convergence.

We summarize an overview of LC-SAC approach in Algorithm 1. Note that, the off-policy RL batch $\mathbf{B}_{rl}$ for training value and policy networks are uniformly sampled from the

entire RL replay buffer $\mathcal{D}_{rl}$, however the context batch $\mathbf{B}_c$ can be uniformly or sequentially sampled from context buffer $\mathcal{D}_c$. In addition, the experience been added to the replay buffer can be $(\mathbf{s}_t, \mathbf{a}_t, r_t)$ or $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}'_t)$.

In section 4, we fist empirically studied the impacts on the performance when removing the next observation in the experience $\mathbf{e}$, then empirically investigated our proposed approach's performance in different sampling strategies and buffer settings in a number of challenging tasks in the continuous control domains.

B. LC-SAC with Diverse-quality Demonstrations

In this part, we incorporate our LC-SAC algorithm with learning from diverse-quality demonstrations, forming a novel approach to effectively learn continuous locomotion skills from both interactions with environments and leveraging existing knowledge (here is human demonstrations), we refer it as LC-SAC-DD.

Some simple modifications on Algorithm 1 can lead to our LC-SAC-DD algorithm.

- First, apart from reusing human knowledge, we still need some interactions to collect information about recent environment and agent. In our setting, at each iteration, N_c steps experiences according to rollout policy $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{c})$ will be added to both the context replay buffer $\mathcal{D}_c$ and the RL replay buffer $\mathcal{D}_{rl}$. Here, the policy is conditioned on the latent context $\mathbf{c}$ sampled from $q_\omega(\mathbf{c}|\mathbf{e})$.
- Second, in collecting demonstrations stage, we directly put the diverse-quality demonstrations into the replay buffer $\mathcal{D}_{rl}$ instead of interacting with environments using concurrent rollout policy.
- Third, in training stage, the context batch $\mathbf{B}_c$ is sequentially sampled from context buffer $\mathcal{D}_c$.

Our training procedure is presented in Algorithm 2. In section 4, we show that the final performence after training by combining demonstrations with the agent's own real experience can outperform the original demonstration used.

Algorithm 2 LC-SAC-DD

Collect diverse-quality demonstrations $\mathcal{D}_{demo}$
Initialize network parameters: $\omega, \theta_{1,2}, \psi, \bar{\psi}, \phi$, and two replay buffer: $\mathcal{D}_c$ for training latent context encoder, $\mathcal{D}_{rl}$ for training policy and value networks

```
while not converged do
  for each collecting data step do
    for t = 0 : N_c do
      sample c ~ q_ω(c|e), rollout policy π_θ(a|s, c) to get experience e_t^c
    end for
    add e^c_{t=1...N_c} to buffer D_c and D_rl
    sample N_rl experiences: e_{t=1...N_rl} from D_demo and add to buffer D_rl
  end for
  for each training step do
    update steps are almost the same as Algorithm 1
  end for
end while
return q_ω, Q_θ, V_ψ, π_φ
```

IV. Experiments

This section presents our experimental results and some analysis. We first present the performance comparison between our proposed LC-SAC over the original SAC algorithm in a collection of continuous tasks. Next, we will show LC-SAC can be incorporated with different quality demonstrations effectively and simply. To be consistent with previous work, our implementation almost use the same network architecture and hyper-parameters across all the tasks except the total training steps. [2]

Experimental Setup In this section, our experiments are based on five robotic locomotion tasks with the MuJoCo simulator [22] [23], namely Ant-v2, HalfCheetah-v2, Hopper-v2, and Humanoid-v2, Striker-v2. Among these tasks, the goal of the robot is to move forward as fast as possible and without falling to the ground. To implement and train the model, we used PyTorch machine learning framework and applied Adam optimization [24] for learning the neural network parameters. Each value and policy network has 3 fully connected layers of 300 hidden units with ReLU non-linear functions.

Latent Context Encoder The latent context encoder network also has 3 fully connected layers of 200 units with ReLU non-linear functions. And we set the dimension of the latent context variable $\mathbf{c}$ as 5. Correspondingly, the output dimension of the latent context encoder network is $D_c{*}2 = 10$, five values in which represents the mean μ of $\mathbf{c}$, five values represent the variance σ of $\mathbf{c}$. The latent context $\mathbf{c}$ is sampled from the Gaussian distribution $\mathcal{N}\left(q^\mu_\omega(\mathbf{e}), q^\sigma_\omega(\mathbf{e})\right)$, where q_ω denotes latent context encoder network,

$$\begin{gathered}\mathbf{q}_\omega(\mathbf{e}, \xi) = \mathcal{G}\left(\mu_\omega(\mathbf{e}) + \sigma_\omega(\mathbf{e}) \odot \xi\right) \\ \text{where } \xi \sim \mathcal{N}(0, I)\end{gathered} \tag{8}$$

Note that, we set memory *Horizon* as 20 or 100. It means the H in $q_\omega(\mathbf{c}|\mathbf{e}_{1:H}) \propto \prod_{t=1}^{H} \mathcal{G}_\omega(\mathbf{c}|\mathbf{e}_t)$, which is adapted from [8]. When the present time step t is less than H, we just use currently collected experience so far, namely $\mathbf{e} = \mathbf{e}_{1:t}$, which could theoretically lead some deviation but empirically, we found it didn't make much harm on final performance.

In the following graphs, each algorithm's performance is measured once every 4000 or 1000 training steps by running the rollout policy for ten trajectories, in different random seeds, and reporting the average return over those test trajectories. More detail experimental hyper-parameters and MuJoCo environment information can be found in Appendix.

A. LC-SAC

We first investigated the effect of the next observation in experience $\mathbf{e}_t$ on our proposed method LC-SAC, in Hopper-v2 environment. Concretely, $\mathbf{e}_t = (\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}'_t)$ or $\mathbf{e}_t = (\mathbf{s}_t, \mathbf{a}_t, r_t)$. The result is shown in Figure 1. From it, we

[2]https://github.com/puyuan1996/IJCNN2020

can get the insight that the next observation is typically not beneficial to the final performance and explicitly increase the computational cost. Thus, we use the $\mathbf{e}_t = (\mathbf{s}_t, \mathbf{a}_t, r_t)$ in our following experiments.

Fig. 1: Using or no using next observation in experience $\mathbf{e}_t$. We compare our LC-SAC in this two variant settings, blue line denotes the performance curve of no using the next observation in $\mathbf{e}_t$, orange line denotes using. We found that no using the next observation in experience is slightly better than using, and is obviously computational cheaper.

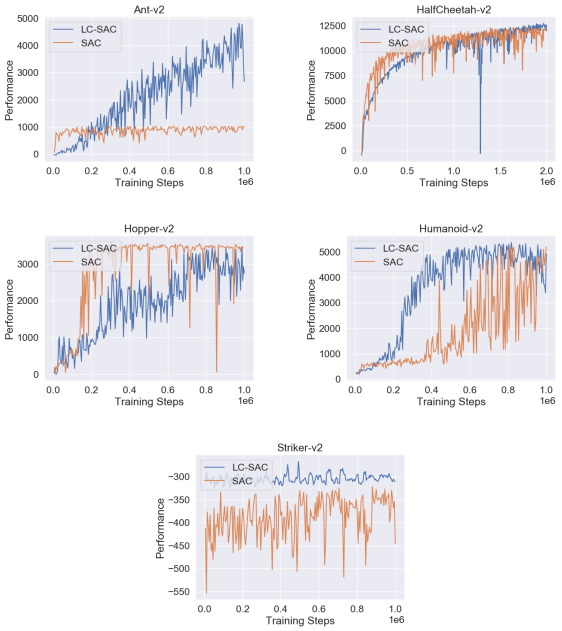

Fig. 2: The performance curves of **LC-SAC** (the context batch $\mathbf{B}_c$ is uniformly sampled from context buffer $\mathcal{D}_c$, the training steps each iteration N_{train}=4000, *Horizon*=100) against original SAC on continuous control tasks. Introducing latent context can offer performance benefits in some tasks.

Next, we assess the performance of our algorithm 1, namely LC-SAC, and analyze its properties comparing against prior original SAC, in the aforementioned continuous tasks. We used the SAC implementation from OpenAI SpinningUp with the same hyper-parameters as the LC-SAC. The plots is showed in Figure 2. We found that our approach obtained comparable final performance with SAC in HalfCheetah-v2, Hopper-v2, learned slightly faster in Humanoid-v2, and outperforms SAC marginally in Ant-v2. We also found both LC-SAC and SAC algorithms cannot solve Striker-v2 task (large than -150 score means been partially solved), but seemly LC-SAC can always obtain 50 more scores than SAC.

LC-SAC-Seq We also evaluated the performance of LC-SAC when the context batch $\mathbf{B}_c$ is sequentially sampled from context buffer $\mathcal{D}_c$. And RL replay buffer $\mathcal{D}_{rl}$ are exactly same as the context replay buffer $\mathcal{D}_c$, in which experiences are collected by roll-outing the policy $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{c})$, and the latent context $\mathbf{c}$ is obtained by sampling from posterior probability $q_\omega(\mathbf{c}|\mathbf{e})$. For brevity, we denote this variant of Algorithm 1 as LC-SAC-Seq below. And the memory *Horizon* is set as 20. The training steps each iteration N_{train}=4000. The performance curves are displayed in Figure 3. (the results when N_{train}=1000 can be found in Appendix Figure 5)

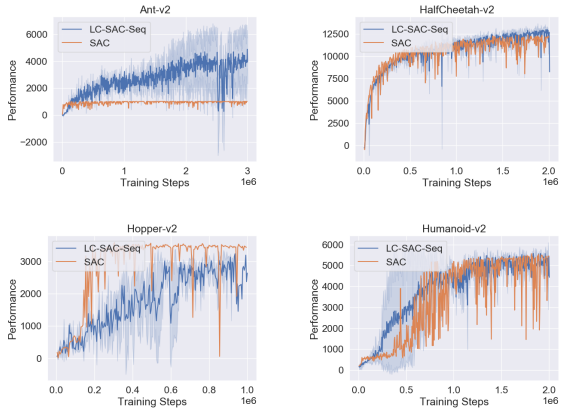

Fig. 3: The performance curves of **LC-SAC-Seq** (the training steps each iteration N_{train}=4000, *Horizon*=20) against original SAC on continuous control tasks. The shaded regions denote the standard deviation in 3 random seeds runs.

In conclusion, both the LC-SAC and LC-SAC-Seq is an effective way to extend the memory of the agent, but at the same time they also induce some bias like certain noise on current state and increase the computation cost. From the experimental results, we see that in some relatively simple tasks (e.g., Hopper-v2), introducing latent context cannot offer performance benefits and even makes the performance of original algorithm worse but not much.

B. LC-SAC-DD

1) Demonstrations: Because lacking the human demonstrations in MuJoCo tasks, to generate diverse-quality demonstrations, we reused the pre-trained Proximal Policy Optimization (PPO) policy [4] [14] which is trained with the ground-truth reward for 500 training steps (64,000 simulation steps) and checkpointed every 5 training steps.

For each checkpoint policy, we generated 200 episodes which have different trajectory length. This provides us diverse quality demonstrations whose performance are almost ranked based on the training stage. In the LC-SAC-DD procedure, we added the demonstrations into the replay buffer according to the checkpoint time of the corresponding policy.

2) Results: We evaluate the performance of LC-SAC-DD on the aforementioned 3 challenging tasks with the continuous action space: Ant-v2, HalfCheetah-v2, Hopper-v2, developed in the MuJoCo physics simulator.

The LC-SAC-DD's performance is showed in the Figure 4 (the results when *Horizon*=100 can be found in Appendix Figure 6). To clearly evaluate the quality of our training approach, we also plot the average return of the original demonstrations by the ordering of training stage of the policy used to generate them and the LC-SAC-DD's performance in the same figure.

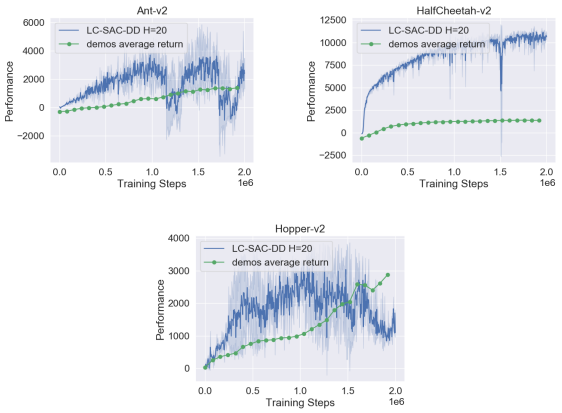

Fig. 4: The performance curves of **LC-SAC-DD** (The training steps each iteration N_{train}=4000, *Horizon*=20) on continuous control tasks. The shaded regions denote the standard deviation in 3 random seeds runs. The green dots denote the average return of the demonstrations sampled to put into the RL replay buffer $\mathcal{D}_{rl}$, at that training step.

When the training steps are 2e6, the required quantity of interacting with environments is $N_c * 500 = 400 * 500 = 2e5$, which is one order of magnitude less than training without human demonstrations. But the performance curves show certain unstable behaviors, especially in Ant-v2, partially because the actor and critic are trained with nearly fullly off-policy data induced by the diverse-quality demonstrations. Even though, we consider that LC-SAC-DD could be a viable way to achieve acceptable performance when largely interacting with environments is expensive or even unrealistic.

V. Conclusion and Future Works

In this paper, we proposed the latent context based soft actor-critic (LC-SAC) which incorporates the latent context variables with SAC, enabling longer-horizon memory about recent environment dynamics and agent's behavioral information. In empirical evaluations on a variety of continuous benchmarks, we found that LC-SAC can obtain comparable results against SAC. To reuse prior human knowledge, we also proposed the LC-SAC-DD procedure, which simply combined LC-SAC with diverse-quality demonstrations, and achieved moderate results while using the relatively small number of interactions with the environment.

In this paper, we only investigate the effect of adding latent context on the continuous domain. Does this method can be used for discrete action space? How to extend LC-SAC to discrete action domains remains as the valuable future work. We also expect that hyper-parameters such as the dimension of the latent context variable, the length of memory horizon can be automatically adjusted during training.

Acknowledgment

Thanks for insightful discussion with Xin Yao and Shaochen Wang. The research is partially supported by the National Natural Science Foundation of China under grant No.U19B2044 and No.61836011.

References

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

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou and D. Hassabis, "Human level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] D. Silver, A. Huang, C. Maddison, et al. "Mastering the game of Go with deep neural networks and tree search," *Nature* vol. 529, pp. 484–489, 2016.

[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv:1707.06347*, 2017.

[5] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver and D. Wierstra, "Continuous control with deep reinforcement learning," in *International Conference on Learning Representations (ICLR)*, 2016.

[6] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning (ICML)*, 2018.

[7] M. Hausknecht and P. Stone. "Deep recurrent Q-Learning for partially observable MDPs," *arXiv:1507.06527*, 2015.

[8] K. Rakelly, A. Zhou, D. Quillen, C. Finn, S. Levine. "Efficient off-policy Meta-reinforcement learning via probabilistic context variables," in *Proceedings of International Conference on Machine Learning (ICML)*, 2019.

[9] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller, "Learning an embedding space for transferable robot skills," in *International Conference on Learning Representations (ICLR)*, 2018.

[10] C. Finn, K. Xu, S. Levine, "Probabilistic model-agnostic meta-learning," in *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[11] J. S. Albus. "Brains, behavior, and robotics." Peterboro, NH: Byte Books, 1981.

[12] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[13] A.Y Ng, S. J. Russell, et al. "Algorithms for inverse reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2000.

[14] D. S. Brown, W. Goo, P. Nagarajan, S. Niekum. "Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations," in *Proceedings of International Conference on Machine Learning (ICML)*, 2019.

[15] B. D. Ziebart, "Modeling purposeful adaptive behavior with the principle of maximum causal entropy," Carnegie Mellon University, 2010.

[16] D. Wierstra, A. Foerster, J. Peters, and J. Schmidthuber. "Solving deep memory POMDPs with recurrent policy gradients," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2007.

[17] N. Heess, J. J Hunt, T. P. Lillicrap, D. Silver. "Memory-based control with recurrent neural networks," *arXiv:1512*, 2015.
[18] C. Finn, P. Abbeel, and S. Levine. "Model-agnostic meta-learning for fast adaptation of deep networks," *arXiv:1703.03400*, 2017.
[19] A. Gupta, R. Mendonca, Y. X. Liu, P. Abbeel, and S. Levine. "Meta reinforcement learning of structured exploration strategies," in *Advances in Neural Information Processing Systems (NIPS)*, 2018.
[20] S. Kapturowski, G. Ostrovski, W. Dabney, J. Quan, R. Munos. "Recurrent experience replay in distributed reinforcement learning," in *International Conference on Learning Representations (ICLR)*, 2019.
[21] M. Taylor and P. Stone. "Transfer learning for reinforcement learning domains: A survey," in *The Journal of Machine Learning Research*, 2009.
[22] E. Todorov, T. Erez and Y. Tassa. "MuJoCo: A physics engine for model-based control." 2012.
[23] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. "OpenAI gym," *arXiv preprint arXiv:1606.01540*, 2016.
[24] D. P. Kingma, J. Ba. "Adam: A method for stochastic optimization," in *International Conference for Learning Representations (ICLR)*, 2015.

APPENDIX

Parameter Name	Value
leaning rate	3e-4
entropy coefficient	0.05
discount factor	0.99
latent context dimensions	5
context encoder network architecture	(200, 200, 200)
value and policy network architecture	(300, 300, 300)
optimizer	Adam
activation function	ReLU
replay buffer size	1e6
RL batch size	256
kl lambda	0.1
N_{c+rl}	1000
Horizon	100 or 20
N_c	400
N_{rl}	600
training steps each iteration (N_{train})	4000 or 1000

TABLE I: Hyper-parameters

Environment	Observation Dimensions	Action Dimensions
Ant-v2	111 (27 dims non-zero)	8
HalfCheetah-v2	17	6
Hopper-v2	11	3
Humanoid-v2	376 (292 dims non-zero)	17
Striker-v2	23	7

TABLE II: Mujoco Environment Parameters

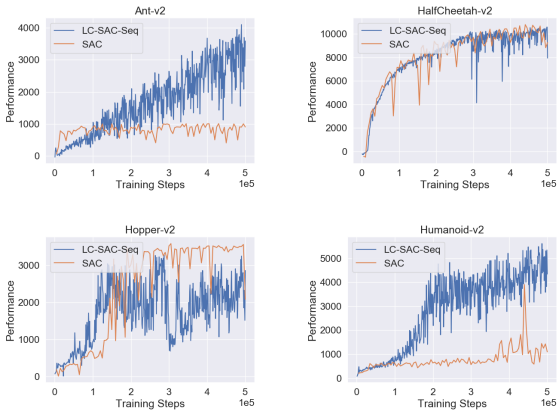

Fig. 5: The performance curves of **LC-SAC-Seq**. (The training steps each iteration N_{train}=1000, *Horizon*=20)

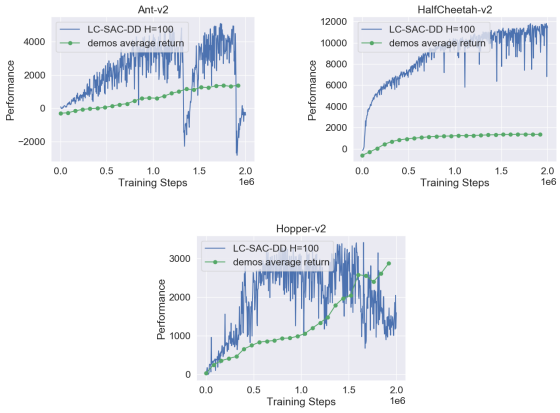

Fig. 6: The performance curves of **LC-SAC-DD**. (The training steps each iteration N_{train}=4000, *Horizon*=100)