HiRL: Dealing with Non-stationarity in Hierarchical Reinforcement Learning via High-level Relearning

Yuhang Jiao,[1] **Yoshimasa Tsuruoka,** [1]

[1] the University of Tokyo
jiao0801@g.ecc.u-tokyo.ac.jp,
yoshimasa-tsuruoka@g.ecc.u-tokyo.ac.jp

Abstract

Hierarchical reinforcement learning (HRL) provides abstraction both on actions and states of the environment to solve tasks with long time horizons and sparse rewards. However, training of hierarchical models suffers the problem of non-stationary environments at the high level since the low-level policy is constantly changing. In this paper, we propose *High-level ReLearning* (HiRL), a novel HRL approach to address this instability problem. First, we introduce a student network to escape the influence of past data. It is trained concurrently with the high-level policy and continuously relearns, and is used to update the original high-level policy. Second, we assist in training student networks via a multi-policy interaction approach. Experimentally, we show that our approach improves performance in several tasks in MuJoCo environments by demonstrating that HiRL can significantly boost the learning performance and exploration capability.

Introduction

Deep reinforcement learning (RL) has achieved great success in solving decision-making problems such as Atari games (Mnih et al. 2013, 2015), chess and cards (Silver et al. 2016), and robot control (Lillicrap et al. 2016; Gu et al. 2017). However, increasing sample efficiency in environments with sparse rewards, solving problems with large action or state space, and performing well in tasks with long time horizons are still a huge challenge for reinforcement learning (Kulkarni et al. 2016; Florensa, Duan, and Abbeel 2017). A promising method to address these issues is Hierarchical Reinforcement Learning (HRL), which decomposes complex tasks into simpler sub-problems. By using a hierarchy of policies in which only the lowest-level policy interacts with the environment, the high-level policy can be trained to plan over a longer time horizon. The actions taken by the high-level policy are used as goals for the low-level policy, and the low-level policy is trained with a goal-conditioned reward function to improve the performance of the agent (Nachum et al. 2018; Vezhnevets et al. 2017; Florensa et al. 2018).

Nevertheless, it is difficult to train multiple levels simultaneously because of the non-stationary state transitions caused by the hierarchical structure. In other words, as the low-level policy constantly changes, the same high-level actions taken in the same state at different steps may result in significantly different state transitions and rewards. Fitting mislabeled data will negatively affect the training process and the final performance of the network, which can significantly affect the exploration and learning ability of the agent in RL training. HIRO (Nachum et al. 2018) and HAC (Levy, Platt, and Saenko 2019) perform relabeling of state transitions to address the problem that non-stationary data can cause instability in subsequent policy learning. However, the relabeling approach does not eliminate the negative effect of old data on the exploration ability of the agent.

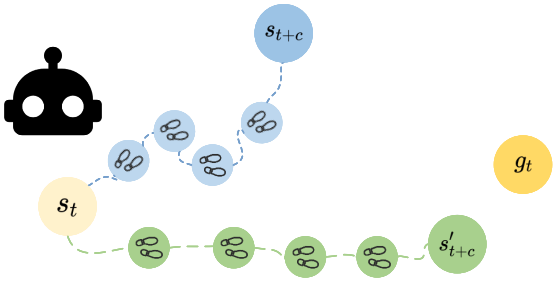


Figure 1: Non-stationarity problem in HRL and its impact on HRL training. It shows the reason for the emergence of the non-stationarity problem. The same goal g_t is given in the same state s_t, with different next states and received rewards due to changes in the low-level policy.

We develop a novel HRL approach named High-level policy ReLearning (HiRL) to tackle this problem. First we analyze the effect of instability of high-level dynamic information and why the relabel method does not completely eliminate this effect. Second, we introduce a student network that is identical to the high-level network to relearn the relabeled high-level state transition data, which reinitializing the student policy erases the memory of the previous data in the parameters that would be hard to remove via finetuning. In the student policy learning phase, we distill the original high-level policy to the student policy to accelerate the training of the student network, utilizing the trained student network to update the parameters of the high-level network at inter-

vals and then re-initialize the student network. To assist the training process of the student network, we use multi-policy interaction to select goals, i.e., to allow the student network to interact with the lower-level policy. We have evaluated our approach in several MuJoCo (Duan et al. 2016) environments. These environments require the ability to perform exploratory navigation and complex sequences of interactions with objects in the environment. Experimental results show that our method exhibits better performance compared to previous HRL algorithms.

Background

Reinforcement learning and MDPs

We assume that the interaction between agent and environment can be modelled as a *Markov decision process* (MDP). An environment is described by a set of states S, a set of actions A, a distribution of initial states $p(s_0)$, a reward function $r : S \times A \to \mathbb{R}$, transition probabilities $p(s_{t+1}|s_t, a_t)$, and a discount factor $\gamma \in [0, 1)$, so we refer to this MDP as $M \doteq (S, A, r, p, p_0, \gamma)$. A policy $\pi(a_t|s_t)$ maps current state s_t to a probability distribution over actions $\pi : S \to A$. Every episode starts with sampling an initial state s_0. According to the policy π, the agent takes an action $a_t \sim \pi(s_t)$, and then the agent enters into a new state s_{t+1} and receives a reward $r_t = r(s_t, a_t)$. A discounted sum of future rewards is called a *return*: $R_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_i$, the objective of the agent is to maximize its expected return $\mathbb{E}_{s_0}[R_0 \mid s_0]$.

Goal-based HRL

We consider a goal-conditioned MDP defined as a tuple $(\mathcal{S}, \mathcal{G}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is a state set, $\mathcal{G}$ is a goal set, $\mathcal{A}$ is an action set, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a state transition, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function, and γ is a discount factor.

We follow the two-level goal-conditioned off-policy hierarchy presented in HIRO (Nachum et al. 2018). The high-level controller with policy $\pi^h_{\theta_h}(g|s)$ aims to maximize the external reward and generates a high-level action, i.e., a subgoal $g_t \sim \pi^h_{\theta_h}(g|s_t) \in \mathcal{G}$, every c time steps, i.e., when $t \equiv 0 \ (mod\ c)$, where $c > 1$ is a pre-determined hyper-parameter. It modulates the behavior of the low-level policy $\pi^l_{\theta_l}$ by intrinsic rewards. The low-level controller aims to maximize the intrinsic reward provided by the high-level controller, and performs a primary action $a_t \sim \pi^l_{\theta_l}(a|s_t, g_t) \in \mathcal{A}$ at every time step. The reward function of the high-level policy is defined as: $r^h_t = \sum_{i=t}^{t+c-1} R(s_i, a_i)$ which is the accumulation of the external reward in the time interval $[t, t + c - 1](t \equiv 0 \ (mod\ c))$, and the intrinsic reward that describes subgoal-reaching performance based on the distance between the current observation and the goal, defined as:

$$r^l_t(s_t, g_t, a_t, s_{t+1}) = -||s_t + g_t - s_{t+1}||_2. \quad (1)$$

State transition relabeling Off-policy HRL suffers the high-level non-stationarity problem since the low-level policy is constantly changing. To solve this problem, HIRO (Nachum et al. 2018) proposes an off-policy correction method. The high-level action g_t which in the past induced a low-level behavior $a_t : t+c-1 \sim \mu^{lo}(s_{t:t+c-1}, g_{t:t+c-1})$ may be re-labeled to a goal $\widetilde{g}_t$ which is likely to induce the same low-level behavior with the current lower-level policy. HAC(Levy, Platt, and Saenko 2019), however, uses the idea of HER(Andrychowicz et al. 2017) to modify the state transitions by taking the state arrived at the low-level after step c as the goal given by the high-level, which satisfies that the low level policy is always near-optimal, thus alleviating the non-stationarity brought by the hierarchical structure.

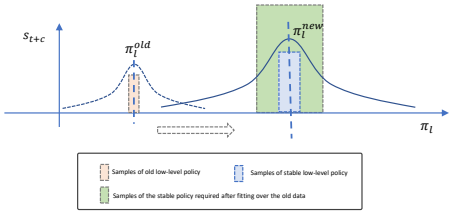


Figure 2: The impact of changing dynamic information on high-level. continual learning (the standard way) is not optimal since the model has been trained using old data and it makes parameters of the model are not suitable for training after that.

High-level Relearning Hierarchical Reinforcement Learning

In this section, we present our framework for addressing the problem of non-stationarity in HRL, HiRL: High-level policy ReLearning. We first analyze the impact of changing dynamic information on high-level learning. Then, we introduce a continuously relearning student policy and subsequently update the parameters of the high-level policy. Since this approach focuses on exploiting experience replay and depends on the exploration of effective state transitions, we propose a simple interaction strategy that does not introduce a new network.

The impact of changing dynamic information on high-level

We consider a standard reinforcement learning setup consisting of an agent interacting with an environment E in discrete time steps. For the estimation of the Q value, many approaches in reinforcement learning make use of the recursive relationship known as the Bellman equation:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{s_{t+1} \sim E, a_t \sim \pi}[r(s_t, a_t) + \gamma \mathbb{E}_{\pi}[Q^{\pi}(s_{t+1}, a_{t+1})]] \quad (2)$$

In a stable environment, the next state s_{t+1} is only environment-dependent and transition-fixed, so it is often ignored when solving for expectations. But in the high-level training of HRL, this dynamic information of environment E is related to the low-level policy and keeps changing. When solving for the expectation, more sampling is required to learn the Q value that is appropriate for the current environment. We can approximate the state after step c as a

Gaussian distribution as the low-level policy tends to stabilize. Then the fitting of the state transitions generated by the unstable low-level leads to the need for more samples in computing the expectation when it is stable. Relabeling method in HIRO allows more samples in the replay buffer to fit the current low-level policy, but because the influence of past data is not removed, it still requires more samples for the expectation calculation when the low-level policy changes. If there is a network that has not been fitted with previous data, the expectation calculation can be performed faster and more accurately, and thus a more accurate value evaluation can be obtained.

In addition, because of the capability of the initial low-level policy, after c steps, the next state s_{t+c} of the high level, often proximity s_t. If it is a negative reward setting, it will lead to a large difference in Q values of the two proximity states, thus making the Q plane sharp and multi-peaked. The relabeling method, however, does not eliminate the influence of the state transitions on the learning process at the beginning, so in the experimental part we can see that the relabeling method is more sensitive and does not perform robustly and well in difficult exploration tasks.

High-level policy Relearning

The non-stationarity in HRL stems from the fact that the dynamic information between the past data and the new data is different in training using experience replay, also, a network with initial training problems can greatly affect the ensuing exploration and thus the learning of the policy throughout the process. HiRL avoids this undesirable effect of poor initialization training by constantly using data that better matches the current low level to find a better high level.

HiRL first performs normal HRL (like HIRO) training to collect trajectory data and later trains the student network using these relabeled high-level state transition data. At the end of the student network training phase, the learned student policy is used to update the original high-level policy in HRL to overcome the non-stationarity in HRL. When learning a new student network, we require the new policy to be unaffected by the previous state transition data. Simultaneously, when the low-level policy tends to stabilize, we need the original high-level policy to instruct the student network to accelerate and stabilize the student training process (see Appendix).

As with HIRO (Nachum et al. 2018), we use the TD3 (Fujimoto, Hoof, and Meger 2018) algorithm for each level in this HRL structure. The actor π^s and critic Q^s of the student are parameterized by θ_{π^s}, θ_{Q^s}, respectively. For the TD3 algorithm, the objective of the actor is to maximize the expected return:

$$L_{PG} = \mathbb{E}_{s_t \sim D}[Q^s(s,a) \mid_{s=s_t, a=\pi^s(s_t|\theta_{\pi^s})}]. \tag{3}$$

where D is the replay buffer after the relabeling of the high level policy. The relabeling method is same as HIRO(Nachum et al. 2018): re-labeling the high-level transition $(s_t, g_t, \sum R_{t:t+c-1}, s_{t+c})$ with a different high-level action $\widetilde{g}_t$ chosen to maximize the probability $\mu^{lo}(a_{t:t+c-1}|s_{t:t+c-1}, \widetilde{g}_{t:t+c-1})$, and the log probability can be computed as:

$$\begin{aligned} &\log \mu^{lo}(a_{t:t+c-1} \mid s_{t:t+c-1}, \widetilde{g}_{t:t+c-1}) \\ &\qquad \propto -\frac{1}{2}\sum_{i=t}^{t+c-1} ||a_i - \mu^{lo}(s_i, \tilde{g}_i)||_2^2 + const \end{aligned} \tag{4}$$

When the low-level is approximately stationary, to accelerate the learning of student actor while simultaneously stabilizing the training process, we introduce a supervised loss L_a to minimize the disagreement between outputs of the student π^s and the original high-level policy π^h:

$$L_a = \mathbb{E}_{s_t \sim D}[(\pi^h(s_t) - \pi^s(s_t))^2]. \tag{5}$$

Combining the two previous terms, we propose the following objective to maximize with θ_{π^s}:

$$\nabla_{\theta_{\pi^s}} J_\pi = \nabla_{\theta_{\pi^s}} L_{PG} - \alpha_\pi \nabla_{\theta_{\pi^s}} L_a \tag{6}$$

where α_π is a variable hyper-parameter from 0 to 1 that we use to control the degree of the constraint of the original high policy on the student policy.

Regarding the optimization objective of the student critic, we use TD learning to fit the Q values. On the other hand, we want the student to approximate the high-level critic to accelerate the training of the student network when the high-level has already learned near accurate Q values.

$$\begin{aligned} L_{TD} =& \mathbb{E}_{s_t, a_t, r_t, s_{t+1} \sim D} \\ & [(Q^s(s_t,a_t) - r(s_t,a_t) - \gamma Q^s(s_{t+1}, \pi^s(s_{t+1})))^2]. \end{aligned} \tag{7}$$

$$L_Q = \mathbb{E}_{s_t, a_t \sim D}[(Q^h(s_t,a_t) - Q^s(s_t,a_t))^2]. \tag{8}$$

Combining the above two items, we can obtain the minimization objective for the student critic:

$$\nabla_{\theta_{\pi^s}} J_Q = \nabla_{\theta_{\pi^s}} L_{TD} + \alpha_Q \nabla_{\theta_{\pi^s}} L_Q \tag{9}$$

where α_Q is a hyper-parameter. Because of the limitations of the low-level capability, the goal previously taken by the high level has a lower corresponding Q value due to the low rewards received. However, this does not mean that this goal is a bad one for the current low-level policy. To eliminate the effect of incorrect Q values, we initially use TD learning to learn the critic, which means that α_Q is equal to 0. Subsequently, the value of α_Q increases gradually. After the student network has completed the training, we perform parameter updates for the high level in the HRL:

$$\theta_{\pi^h} \leftarrow \beta\theta_{\pi^s} + (1-\beta)\theta_{\pi^h} \tag{10}$$

$$\theta_{Q^h} \leftarrow \beta\theta_{Q^s} + (1-\beta)\theta_{Q^h} \tag{11}$$

β is a hyper-parameter that controls the degree of update of the high-level networks, with the value of β gradually decreasing as the high-level training approaches stationary.

Multi-policy interaction

Because the student network's training relies on the state transitions collected by the original policy, it is difficult for the student network to update the values of states that are not explored by the original policy and some states that need

Algorithm 1: Pseudo-Code for HiRL

Require: Initial training phase t_{init}, Student training phase $t_{student}$

Initialize actor π^h,π^l, critic Q^h,Q^l and memory D
for t_{init} steps **do**
  Update π^h,π^l,Q^h and Q^l using the standard RL training using D
end for
while True **do**
  Initialize student actor π^s, critic Q^s;
  for $t_{student}$ steps **do**
    Collect transitions using multi-policy exploration method
    Update π^h,π^l,Q^h and Q^l using the standard RL training
    Update π^s and Q^s with Eq. 6 and Eq. 9 using the relabeled state transitions experience in D
  end for
  Update π^h and Q^h with Eq. 10 and Eq. 11 using π^s and Q^s
  Discard π^s and Q^s
end while

to be re-explored. Such a dataset is uncorrelated to the true distribution under the current policy, and the phenomenon is called extrapolation error (Fujimoto, Meger, and Precup 2019). Furthermore, assuming that the original policy falls into the local optimum, the extreme lack of diversity of state transition data will further exacerbate the difficulty of policy learning. We therefore propose a simple method that allows the student network to join the interaction with the low level.

$$g_t = \begin{cases} \pi^h(s_t), & 0 < p < p_{threshold} \\ \pi^s(s_t), & p_{threshold} < p < 1 \end{cases} \tag{12}$$

In the student training phase, with probability $1 - p_{threshold}$, the low-level receives a goal from the student, and with probability $p_{threshold}$, the low-level receives a goal from the high level policy. Because the need to train a scratch student network using data collected at a high level creates an extrapolation error arising from the large difference in the distribution of data, involving the student in the interaction with the low-level can mitigate this error to some extent. In HAC(Levy, Platt, and Saenko 2019), they also keep part of the original state transitions without relabeling. Because the action distribution after relabel differs significantly from the actual policy action distribution, which is not conducive to policy training.

Experiments

This section details the experimental setup and training procedure and presents the results of various continuous control tasks. We design the experiments to answer the following research questions: (1) Does HiRL improve the robustness against the relabeling method? (2) Does HiRL improve the performance against other end-to-end HRL algorithms? (3) What role does each part of the algorithm play?

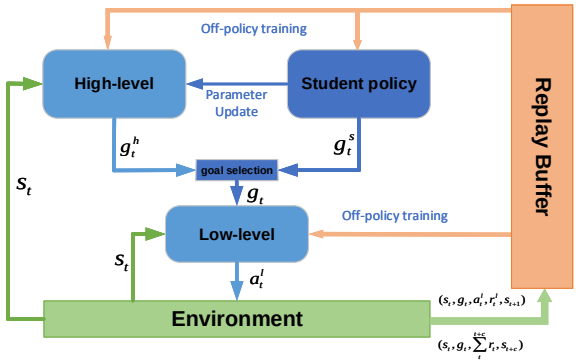


Figure 3: The design and training process of HiRL. The low-level policy interacts directly with the environment, and the high-level policy generates goals every c steps to guide the low-level policy. We periodically train a student network to assist the high-level policy learning, and the student network is trained using the relabeled state transitions. After each training period, the student network is re-initialized after updating the parameters of the high-level network.

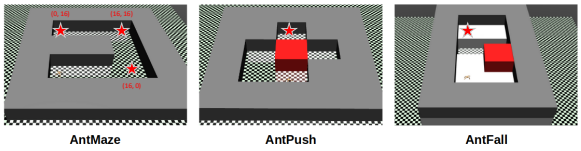


Figure 4: Three hierarchical navigation tasks we consider: Ant Maze, Ant Push, and Ant Fall. The ant is rewarded for approaching the target location (red star). A successful policy must perform a complex sequence of directional movement and, in some cases, interact with objects in its environment (red blocks), e.g., pushing aside an obstacle (middle) or using a block as a bridge (right). A high-level policy periodically produces goal states (corresponding to desired positions and orientations of the ant and its limbs) that the low-level policy tries to match.

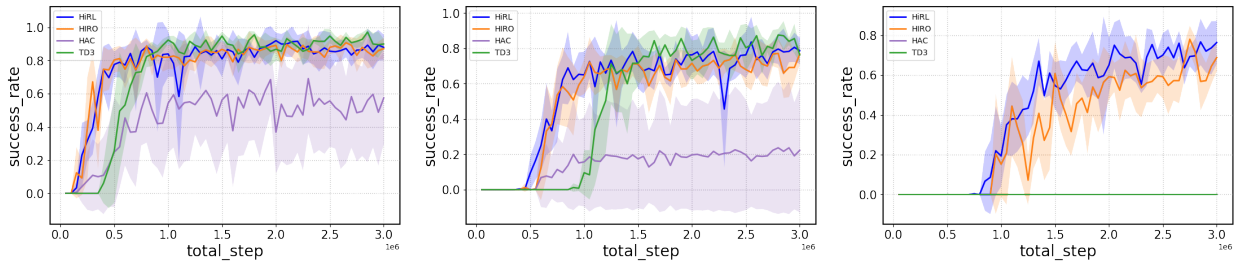
Environmental Setup

The performance of the proposed method is evaluated on a set of long-horizon control tasks. We evaluate and analyze our methods in the benchmark hierarchical tasks, which were all simulated using the Mujoco environment:

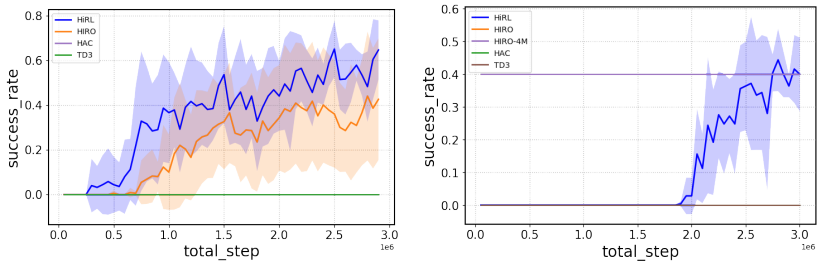
Ant Maze. In this task, the ant agent is placed in a U-shaped corridor with an initial position (0,0) and assigned an (x, y) goal position in the corridor. The agent is rewarded with its negative L_2 distance from the current position to this (x, y) goal position.

Ant Push. This task has a movable block that needs to be pushed by an ant, which exists on the path from the initial position to the target position. The ant needs to move around to the left of the block and then push the block to the right to reach the target position.

Ant Fall. This task extends navigation to three dimen-



(a) AntMaze(16,0). (b) AntMaze(16,16). (c) AntMaze(0,16).



(d) AntPush. (e) AntFall.

Figure 5: Evaluation results in the AntMaze, AntPush, and AntFall. 50 randomized evaluation experiments are conducted every 50,000 steps and the success rate is calculated as the evaluation criterion.

sions. The ant is placed on an elevated platform with the target location directly in front of it but separated by a chasm that cannot be traversed. Fortunately, a movable block is provided on its right side. The ant must first walk to the right, push the block into the chasm, and cross safely to reach the target successfully.

Comparative Analysis

We test our proposed algorithm against the following baseline methods:

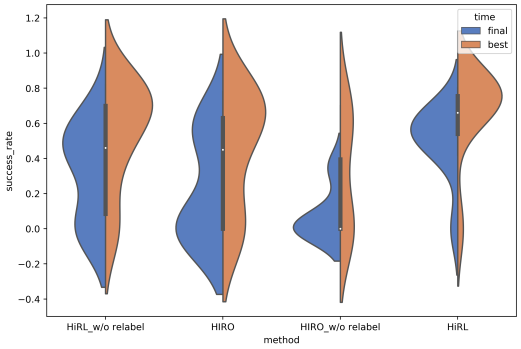
TD3 (Fujimoto, Hoof, and Meger 2018): a state-of-art flat RL algorithm, we chose it for comparison to validate the need for goal-conditioned hierarchies.

HIRO (Nachum et al. 2018): a state-of-art off-policy HRL algorithm using the off-policy correction to address the non-stationarity.

HAC (Levy, Platt, and Saenko 2019): a state-of-art off-policy HRL algorithm using the hindsight relabeling method to address the non-stationarity.

We evaluated the results over 10 randomly seeded experiments, where the time horizon is set to 500 steps. The algorithmic framework and parameter settings are kept consistent with HIRO to estimate the difference between the relearning and relabel methods. Both levels employ TD3. The low-level and high-level critics are updated every step and every 10 steps, respectively. The low-level and high-level actors are updated every 2 steps and every 20 steps, respectively.

To answer the first question, we perform a comparison in Antpush environmental task, with ten random seed experiments, and it is easy to see from Figure 6 that HiRL performs more robustly across multiple experiments after



(a) The performance of different methods on the AntPush task

Figure 6: Non-stationarity problem in HRL and its impact on HRL training. It shows the success rate distribution of the four methods on the AntPush task, **final** represents the final performance of the agent in each experiment, and **best** is the best performance achieved by the agent in each single experiment. The **best** can reflect the exploration ability, and if it is 0, it indicates that the agent is trapped in the local optimum. The gap between the **best** and the **final** can reflect the stability of the single experiment training, and the top-bottom difference illustrates the stability of the algorithm between multiple experiments. In the absence of relabeling and relearning, the difference between the best performance and the final performance is more significant. After relabeling, it can be seen that the subsequent policy training has been stabilized, but the exploration is still affected. After the relearning process, the exploration is enhanced.

reinitialization. While the relabeling method stabilizes the training in separate experiments, the performance of the ten experiments varies more in the case of changing only the random seeds, which verifies what we said before.

To answer the second question, we conduct the comparative experiments in three different environments. Notice that we do not have the initial position of the agent cover the entire state space as in the original HAC, which explains the poor performance of the HAC in Figure 5. In the AntMaze, we evaluate the performance of algorithms by the metric of success rate, once every 50,000 steps for the target position (16,0), (16,16), and (0,16), respectively. The results are shown in Figure 5, demonstrating the training performance of HiRL and other baselines. For the target location (16,0), all methods achieve similar performance, as the ant can easily reach the target directly in front of it. For the target locations (16, 16) and (0, 16), the agent should learn to change its direction, HIRO achieves good performance in this task, and HiRL performs slightly better than HIRO. In the AntPush and AntFall, the requirements for the agent are more complex compared to the AntMaze task, and the moveable block makes the task less fault-tolerant, which requires the agent to be more exploratory and capable of stepping out of the local optimum. We can see from Figure 5 that HiRL also exhibits higher performance than the other baselines. We can also see from Figure 5 and 6(a) that HiRL shows a more robust exploration and learning ability than HIRO, with a smaller variance in its performance than HIRO. It indicates that the relabel method cannot effectively eliminate the adverse effect of the non-stationarity problem on exploration while the relearning process can. Notice that in the AntFall task, "HIRO-4M" represents the performance in the original HIRO paper, which achieved a success rate of 0.4 after 4M steps, while HiRL showed a superior exploration ability.

Ablative Analysis

To investigate how each component contributes to the algorithm, we design several variants to conduct experiments. Figures 7 and 8 show the performance of several variants in the same experimental setting.

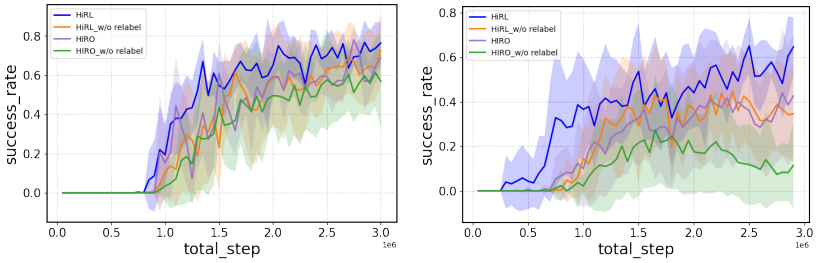
Relearning. To explore the effectiveness of the relearning method in solving the non-stationarity problem, we evaluated the performance of the algorithm when only the relearning method is used. With relearning only, we use the recent part of the replay buffer to avoid the impact of non-stationary data on the training of the student network. As can be seen from Figures 7(a) and 7(b), relearning can mitigate the instability problem even though the data are not relabeled. Due to the unrelabeled data, however, the available replay buffer size is limited and the performance of the algorithm is affected. See Appendix for more details.

Multi-policy interaction. To examine the necessity for multi-policy interaction, we remove the interaction of student network participation with the lower-level policy. We compare the performance of the multi-policy interaction method in assisting student network training and improving algorithm performance, respectively. As we illustrated in Section , direct off-policy learning by the student using data from the experience replay buffer is not effective. Figure 8(a) demonstrates that the multi-policy method effectively enriches the diversity of sampled data and facilitates the learning of the student policy. Figure 8(b) illustrates that involving the student network in the interaction not only assists in network training (the training process becomes less volatile and there are no more plunges when the student network replaces the higher levels), but also increases the exploration ability of the agent and improves learning performance.

Related Work

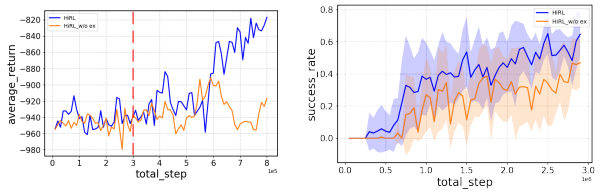
HRL is effective in dealing with long-period and sparse reward problems. Intuitively, HRL is (at least) a two-level approach, where the high-level policy decomposes the problem into subtasks. In contrast, the low-level policy learns how to solve these subtasks efficiently. The exact details of the decomposition, and the specific way the higher level communicates with the lower level varies between approaches. Thus, the signal sent to the lower levels can be some discrete value for selecting options (Bacon, Harb, and Precup 2017; Fox et al. 2017; Gregor, Rezende, and Wierstra 2017) or skills (Konidaris and Barto 2009; Eysenbach et al. 2019; Sharma et al. 2020), or it can be a continuous vector that sets a subgoal in the state space or potential space (Vezhnevets et al. 2017). One way of temporal abstraction is for the high level to give goals that need to be accomplished by the low level. Generally, off-policy HRL algorithms are more sample-efficient than on-policy algorithms. However, because the low-level policy is constantly changing, this non-stationarity becomes particularly evident in off-policy training. HIRO (Nachum et al. 2018) and HAC (Levy, Platt, and Saenko 2019) relabel past trajectories to reduce the impact when inappropriate state transitions are sampled during training. Another direction to address this non-stationarity problem is to stabilize the training process by reducing improper state transitions. HRAC (Zhang et al. 2020) reduces the number of changeable state transitions by calculating the shortest transition distance to generate k-step adjacent goals that are close enough so that the low-level policy can reach the specified state. In this way, the same state (i.e., the goal state) can still be reached even after the high level gives the same goal action in the later stages of training. The non-stationarity due to the interaction between agents in the multi-agent system has similarities with the non-stationarity due to the interaction between different levels in HRL. I^2HRL (Wang et al. 2020) proposes enhancing the communication between different levels to stabilize the training process. Faced with a time-varying dynamic environment, SWUCRL2-CW (Cheung, Simchi-Levi, and Zhu 2020) and RL-CD (Da Silva et al. 2006) determine the environmental changes by detecting the maximum difference between transition probability functions. Once a change in the environment is detected, the estimation restarts. ITER (Igl et al. 2021) uses policy distillation to learn a sequence of policies to overcome the non-stationarity due to policy enhancement.

Inspired by these approaches, we propose a method based on restart and policy distillation to continuously improve the



(a) AntMaze(0,16). (b) AntPush

Figure 7: Evaluation results in the AntMaze and AntPush. We compare the four methods of original HRL (HIRO w/o relabel), relabel only (HIRO), relearning only (HiRL w/o relabel), and relabel+relearning (HiRL). Notably, the non-stationarity problem is improved even when only relearning is performed.



(a) training of student network (b) AntPush

Figure 8: Ablation experiments. HiRL represents the algorithm that uses multi-policy interaction policy, HiRL-w/o ex represents the relearning method that removes the interaction policy. The left figure (a) shows the learning effect of the student network under two approaches. The student network starts training after the red line. The training uses only the normal TD3 optimization objective without introducing supervisory loss, the results are compared in Appendix after introducing supervised loss. The right figure (b) shows the improved performance of the algorithm after the introduction of multi-policy interaction method.

effectiveness of the original policy network by constantly relearning a student network to eliminate the effect of previous state transitions on the learning process.

Conclusion

In this work, we focus on the non-stationarity problem in HRL. We find that the impact of old data is negative for subsequent network training and devastating for agent exploration. We introduce a continuously relearning student network to help the high-level policy eliminate the adverse effect of the old data to address this problem. Through experiments, the results show that HiRL does have the ability to help the agent alleviate the influence of old data and increase the exploration ability of the agent. Also, our experimental results suggest that this problem can be summarized as a result of overfitting the high-level network to the old data, so whether there is a simple regularization method to attenuate this fitting problem deserves further study.

Limitations

We have improved the non-stationarity problem in HRL by introducing a student network. Although better performance and more robust learning over multiple experiments are achieved, the additional network and the new hyperparameters increase the burden of computational resources for training and the work of tuning the hyperparameters. Additionally, we have tested in several typical environments, but there is still a need to test in more environments.

References

Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Abbeel, P.; and Zaremba, W. 2017. Hindsight experience replay. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 5055–5065.

Bacon, P.-L.; Harb, J.; and Precup, D. 2017. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Cheung, W. C.; Simchi-Levi, D.; and Zhu, R. 2020. Reinforcement learning for non-stationary Markov decision processes: The blessing of (more) optimism. In *International Conference on Machine Learning*, 1843–1854. PMLR.

Da Silva, B. C.; Basso, E. W.; Bazzan, A. L.; and Engel, P. M. 2006. Dealing with non-stationary environments using context detection. In *Proceedings of the 23rd international conference on Machine learning*, 217–224.

Duan, Y.; Chen, X.; Houthooft, R.; Schulman, J.; and Abbeel, P. 2016. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, 1329–1338. PMLR.

Eysenbach, B.; Gupta, A.; Ibarz, J.; and Levine, S. 2019. Diversity is All You Need: Learning Skills without a Reward Function. In *International Conference on Learning Representations*.

Florensa, C.; Duan, Y.; and Abbeel, P. 2017. Stochastic neural networks for hierarchical reinforcement learning. In *International Conference on Learning Representations*.

Florensa, C.; Held, D.; Geng, X.; and Abbeel, P. 2018. Automatic goal generation for reinforcement learning agents. In

International conference on machine learning, 1515–1528. PMLR.

Fox, R.; Krishnan, S.; Stoica, I.; and Goldberg, K. 2017. Multi-level discovery of deep options. *arXiv preprint arXiv:1703.08294*.

Fujimoto, S.; Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 1587–1596. PMLR.

Fujimoto, S.; Meger, D.; and Precup, D. 2019. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, 2052–2062. PMLR.

Gregor, K.; Rezende, D. J.; and Wierstra, D. 2017. Variational intrinsic control. In *International Conference on Learning Representations*.

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

Igl, M.; Farquhar, G.; Luketina, J.; Boehmer, W.; and Whiteson, S. 2021. Transient Non-stationarity and Generalisation in Deep Reinforcement Learning. In *International Conference on Learning Representations*.

Konidaris, G. D.; and Barto, A. G. 2009. Efficient Skill Learning using Abstraction Selection. In *International Joint Conference on Artificial Intelligence, IJCAI*, volume 9, 1107–1112. Citeseer.

Kulkarni, T. D.; Narasimhan, K.; Saeedi, A.; and Tenenbaum, J. 2016. Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. In Lee, D.; Sugiyama, M.; Luxburg, U.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Levy, A.; Platt, R.; and Saenko, K. 2019. Hierarchical Reinforcement Learning with Hindsight. In *International Conference on Learning Representations*.

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

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing Atari with Deep Reinforcement Learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Nachum, O.; Gu, S. S.; Lee, H.; and Levine, S. 2018. Data-Efficient Hierarchical Reinforcement Learning. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Sharma, A.; Gu, S.; Levine, S.; Kumar, V.; and Hausman, K. 2020. Dynamics-Aware Unsupervised Discovery of Skills. In *International Conference on Learning Representations*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587): 484–489.

Vezhnevets, A. S.; Osindero, S.; Schaul, T.; Heess, N.; Jaderberg, M.; Silver, D.; and Kavukcuoglu, K. 2017. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, 3540–3549. PMLR.

Wang, R.; Yu, R.; An, B.; and Rabinovich, Z. 2020. I2HRL: Interactive Influence-based Hierarchical Reinforcement Learning. In Bessiere, C., ed., *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 3131–3138. International Joint Conferences on Artificial Intelligence Organization. Main track.

Zhang, T.; Guo, S.; Tan, T.; Hu, X.; and Chen, F. 2020. Generating Adjacency-Constrained Subgoals in Hierarchical Reinforcement Learning. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 21579–21590. Curran Associates, Inc.