IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 3, 2019, accepted August 16, 2019, date of publication August 22, 2019, date of current version September 5, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2936863

Rewards Prediction-Based Credit Assignment for Reinforcement Learning With Sparse Binary Rewards

MINAH SEO, LUIZ FELIPE VECCHIETTI, SANGKEUM LEE, AND DONGSOO HAR, (Senior Member, IEEE)

The Cho Chun Shik Graduate School of Green Transportation, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea

Corresponding author: Dongsoo Har (dshar@kaist.ac.kr)

This work was supported by the Institute for Information and Communications Technology Promotion (IITP) grant funded by the Korean Government (MSIT) (Development of Robot Hand Manipulation Intelligence to Learn Methods and Procedures for Handling Various Objects with Tactile Robot Hands) under Grant 2018-0-00677.

ABSTRACT In reinforcement learning (RL), a reinforcement signal may be infrequent and delayed, not appearing immediately after the action that triggered the reward. To trace back what sequence of actions contributes to delayed rewards, e.g., credit assignment (CA), is one of the biggest challenges in RL. This challenge is aggravated under sparse binary rewards, especially when rewards are given only after successful completion of the task. To this end, a novel method consisting of key-action detection, among a sequence of actions to perform a task under sparse binary rewards, and CA strategy is proposed. The key-action defined as the most important action contributing to the reward is detected by a deep neural network that predicts future rewards based on the environment information. The rewards are re-assigned to the key-action and its adjacent actions, defined as adjacent-key-actions. Such re-assignment process enables increased success rate and convergence speed during training. For efficient re-assignment, two CA strategies are considered as part of proposed method. Proposed method is combined with hindsight experience replay (HER) for experiments in the OpenAI gym suite robotics environment. In the experiments, it is demonstrated that proposed method can detect key-actions and outperform the HER, increasing success rate and convergence speed, in the Fetch slide task, a type of task that is more exacting as compared to other tasks, but is addressed by few publications in the literature. From the experiments, a guideline for selecting CA strategy according to goal location is provided through goal distribution analysis with dot map.

INDEX TERMS Credit assignment, delayed rewards, goal distribution, reinforcement learning, reward shaping.

I. INTRODUCTION

Reinforcement learning (RL) refers to a machine learning method that allows an agent to learn actions to achieve goals with minimum supervision by providing reinforcement signal in the form of negative or positive reward. Combined with recent neural network structures, the RL is utilized to obtain breakthrough results in various areas such as game environments [1]–[4], robotic tasks [5], and autonomous driving control [6]. For some type of RL problems, a reinforcement signal does not appear immediately after the action that triggers the reward. In this case, the RL algorithm faces what is called delayed rewards. Handling delayed rewards is one of the biggest challenges in RL. The training performance is insufficient because the definition of which action induces rewards is ambiguous [7]. This problem can be solved by assigning credit to the action that triggers a reward so that the agent can take the future value into account when it chooses an action. This is defined as a credit assignment (CA) problem in a delayed rewards environment [8]. In RL, the CA for a received reward is one of the main tasks when modeling the system. Many off-policy algorithms being used extensively, such as deep Q-networks [1], are methods involving CA with neural networks. They allow an agent to learn a policy even if there exists a delay between an action and the corresponding reward by propagating the reward

The associate editor coordinating the review of this article and approving it for publication was Shuihua Wang.

backward with guaranteed convergence [9]. These algorithms drive the agent to repeat the actions which lead to the highest accumulated reward by updating the value of a specific action in a specific state, called Q value, based on the rewards that the actions receive. The iteration of this process propagates the delayed reward to the action that triggers the reward through Q values of sequential actions between them.

Even with these algorithms, training becomes extremely difficult when sparse binary rewards are used for RL. There are no intermediate rewards, except the reward given in two values, e.g., -1 or 0, when the agent achieves the goal. Sparse binary rewards are free from conventional reward shaping, which is prone to accompany developer's bias and needs domain knowledge to make reward function. However, low frequency of reward for success, or poor sampling efficiency, as well as the delay of reward significantly retards the training [10]. Training in an environment with vast state space such as continuous robot arm control aggravates this problem. Therefore, improving the training performance under sparse binary rewards is a key issue in applying RL algorithm in complex applications.

Our method proposed in this paper is designed to deal with this problem in a high-dimensional RL environment under sparse binary rewards. In general, the CA for RL with sparse binary rewards requires large computing power. On the other hand, effective training is enabled through CA by assigning the delayed reward directly to the action that contributes to the achievement of the goal. Effect of this direct assignment is similar to that of reward shaping. As stated in [11], this results in large savings in training time and, as a result, alleviating poor sample efficiency. The action that contributes most to a reward for success, defined as the *key-action*, and actions before or after the key-action, defined as *adjacent-key-actions*, are detected and new rewards are assigned to these actions. With the presence of adjacent-key-actions also receiving rewards, sampling efficiency increases and as a result success rate and convergence speed are increased.

Among existing methods for CA, reward shaping is representative. Static reward shaping was widely used before dynamic reward shaping has been introduced lately. Static reward shaping uses reward functions whose returns do not change with the agent's experience. Before training, developers create a criteria to provide rewards based on domain knowledge and the criteria is static over training time [12]. Because the states where additional rewards are not imposed are not explored, only the states where rewards are given can be trained effectively [13]. This is called biased learning. Reward functions using static reward shaping should be modified whenever the task or environment changes, so that its application to multiple environments is not easy [14].

Dynamic reward shaping is closely related to the proposed method, considering it uses reward functions whose time-varying returns depend on the experiences. As the learning progresses, the reward function is generated based on the observed experiences during training. In [15], a Bayesian reward shaping method to distribute rewards is proposed. It uses prior beliefs regarding the environment as a form of introduction to shape the rewards. In [16], two algorithms for restructuring reward functions intended to reduce the gap between the rewards and action that causes them are presented. Hybrid reward decomposition in multi-objective tasks is evaluated in [17].

Exploration is a critical part for facilitating CA with neural networks. To improve exploration, effective sampling is important. There have been some ways to improve sampling efficiency. In [18] the episodes stored are split into two memories, one for successful episodes and the other for unsuccessful episodes. To keep a proper rate of successful episodes in a training batch, they are sampled with a fixed probability. In [19], an experience with high expected learning improvement has priority when it is sampled from replay buffer. It utilizes the temporal-difference (TD) error magnitude as a prioritization mechanism. There is another way to improve exploration and sampling efficiency. The hindsight experience replay (HER) algorithm [20] effectively explores the environment by replacing the original goal of an episode with the actual result of the episode. This increases the frequency of the reward for success under sparse binary rewards.

In this paper, a novel method comprising key-action detection and CA strategies is proposed, in order to increase success rate and convergence speed during training in high-dimensional RL environment under sparse binary rewards. For the key-action detection, rewards prediction model is created. With the CA strategies, adjacent-key-actions are determined and then rewards are re-assigned to the key-action and adjacent-key-actions. This reward re-assignment to the key-action and adjacent-key-action(s) enhances success rate and convergence speed during training. The CA strategies designed in this paper are CA_1 strategy, which selects adjacent-key-actions among a series of actions before the key-action, according to the hyperparameter distribution length D and CA_2 strategy, which selects adjacent-key-actions between the key-action and another action assigned a reward for success for the first time after the key-action, according to another hyperparameter offset length O. Underlying assumption on adjacent-key action(s) of the CA_1 strategy is that the set of actions before the key-action is for preparation of the key-action and adjacent-key-action(s) can be considered as a subset of it. Another set of actions after the key-action is considered less likely to contribute to received reward. On the other hand, the CA_2 strategy pays attention to the enhanced exploration of the key-action by increasing the Q value of the adjacent-key-action(s) that cannot be conducted without the key-action.

To verify the performance of the proposed method, the Fetch push and Fetch slide tasks in OpenAI gym suite [21], which is a toolkit for developing and comparing reinforcement learning algorithms, are considered. The goal space for the two tasks is the surface on the table and it is divided into near zone within the reach of robot arm (1.2 m)

and far zone. The two tasks considered in this paper can be reclassified into two categories according to cause and effect relationship between actions and reward. In case of the Fetch push task, the robot arm continually pushes the puck as far as the reach of the robot arm. This type of task where all sequential actions before generation of a reward affect the value of the reward is referred to as *Evenly Distributed chance of Actions to affect Reward* (EDAR) task. The EDAR tasks are general RL tasks, such as solving a maze, lifting a box and moving it to a specific location, and pushing a box to a specific location. In case of the Fetch slide task, the robot arm continually pushes the puck as far as the reach of the robot arm to earn a reward, however, for the goals beyond the reach it hits the puck. Therefore, the Fetch slide task is fulfilled by push action and hit action, which represents combination of different types of actions. The hit action is more appropriately fit to other type of task, namely non-EDAR task. The *non-EDAR* task is a task where some actions among sequential actions before generation of a reward do not affect the value of the reward. Examples of this task include shooting an object to a specific location, passing a ball, and the famous Atari game named *breakout* that breaks the blocks by bouncing a ball with a short bar. In the non-EDAR task, an agent can affect the state of the object only at a specific moment. The Fetch push task and Fetch slide task with goals in near zone, where the goal is achieved more effectively by push action, can be classified into EDAR tasks, whereas the Fetch slide task with goals positioned in far zone, requiring hit action to achieve goals, can be classified into non-EDAR tasks. The Fetch slide task with goals positioned in far zone is particularly called in this paper far-zone Fetch slide task. Main contributions of this paper are as follows.

1) A novel method consisting of key-action detection and two CA strategies is proposed for EDAR and non-EDAR tasks. The non-EDAR task is known to be hard to fulfill, unlike other tasks [22]. A deep neural network is applied to predict rewards in future timesteps. With the predicted rewards, a key-action is defined. The key-action is used to re-assign reward for success in a sparse binary rewards environment. Showcase of two CA strategies for the Fetch push and Fetch slide tasks is presented.
2) Classification rule according to cause and effect relationship between actions and reward is introduced. The EDAR task is a task where all sequential actions before generation of a reward affect the value of the reward. The non-EDAR task is a task where some actions among sequential actions before generation of a reward do not affect the value of the reward.
3) Experiments for comparison of proposed method with the deep deterministic policy gradient (DDPG) [23] combined with the HER, e.g., DDPG+HER, are presented. Comparison of success rate and convergence speed between the proposed method and DDPG+HER is presented. In addition, variation of mean Q value with the proposed method and DDPG+HER is shown.
4) For analysis of success and failure in achieving goals according to goal position, use of two dot maps is introduced. Success dot map represents distribution of achieved goals and failure dot map shows distribution of unachieved goals. It is important to see by an analysis which goals are achievable by the policy and which ones are not, because such characterization enables efficient use of the policy.
5) A guideline for selecting CA strategy according to goal location is provided from the results of experiments. The CA_1 strategy works in near zone better than CA_2 strategy and DDPG+HER, which corresponds to the Fetch push task and Fetch slide task with goals positioned in near zone, whereas the CA_2 strategy performs in far-zone Fetch slide task better than CA_1 strategy and DDPG+HER. Therefore, combined use of CA_1 and CA_2 strategies outperforms the DDPG+HER regardless of goal position.

The rest of this paper is organized as follows. In Section II, concepts of off-policy RL, DDPG, and HER are described. Section III introduces the proposed method in details. Section IV shows the experimental results. Section V presents concluding remarks.

II. SYSTEM MODELING

In this section, the concept and mathematical model of an off-policy RL algorithm, DDPG, and HER are presented.

A. OFF-POLICY REINFORCEMENT LEARNING

In the general RL framework, an agent, represented in this paper by a robot, interacts with an environment to perform a task in a finite number of discrete timesteps. Let $\mathcal{S}$ and $\mathcal{A}$ be the state and action space, respectively. At each timestep t, the agent observes a state $s_t \in \mathcal{S}$ and chooses an action $a_t \in \mathcal{A}$ based on a policy π. The policy π is a function that maps states into actions, and often is approximated by a neural network. As a consequence of taking the action a_t, the agent receives a reward r_t and observes the next state s_{t+1} of the environment. The reward at timestep t is defined by $r_t = r(s_t, a_t)$ and the next state s_{t+1} is sampled according to the transition probability $p(s_{t+1}|s_t, a_t)$. For each environment timestep, an experience (s_t, a_t, r_t, s_{t+1}) is stored in the experience replay buffer until the end of an episode. The framework considers the Markov decision process assumption as the state s_{t+1} is only conditioned by s_t and a_t. The agent interacts with the environment until it reaches the last timestep, called the terminal state.

To calculate the value of a state s_t, a return variable is defined as $R_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_i$, that is the total accumulated return from timestep t with discount factor $\gamma \in [0, 1]$. The goal of the agent is to take actions so as to maximize the expected return for a given state s_t. The Q function or action-value function following a policy π is defined as $Q^{\pi}(s, a) = \mathbb{E}[R_t|s = s_t, a]$ and can be interpreted as the expected return for selecting action a in state s. For an optimal policy π^*, the optimal Q function Q^* satisfies the Bellman equation

given as $Q^*(s_t, a_t) = \mathbb{E}\left[r_t + \gamma \max_{a_{t+1} \in \mathcal{A}} Q^*(s_{t+1}, a_{t+1})\right]$ and can be interpreted as the expected return at timestep t when the action that maximizes the Q function is taken in the next state s_{t+1}.

B. DEEP DETERMINISTIC POLICY GRADIENTS

Like the policy π, the optimal Q function can be approximated by a deep neural network. The neural networks used to approximate the policy and Q function are called the actor and critic networks, respectively. These networks can be trained by a variety of actor-critic RL algorithms. An example of an actor-critic algorithm used for robot control is the DDPG. The DDPG is an off-policy and model-free RL algorithm for continuous action space. In the DDPG, the critic network is trained by minimizing the loss $\mathcal{L}_c = (r_t + \gamma Q(s_{t+1}, \pi(s_{t+1})) - Q(s_t, a_t))^2$ and the actor network is trained by minimizing the loss $\mathcal{L}_a = -\mathbb{E}_s[Q(s_{t+1}, \pi(s_{t+1}))]$.

C. HINDSIGHT EXPERIENCE REPLAY

The HER proposed in [20] attempts to solve the problem of sampling efficiency in tasks where multiple different goals (multigoal) can be achieved. Its main concept comes from the proposition that in these tasks it is possible to learn not only from successful experiences but also from unsuccessful experiences. In multigoal environments, a goal g representing the final goal of the task is added to the input of actor and critic networks. The experience is redefined as $(s_t, g, a_t, r_t, s_{t+1})$. The HER assumes that a mapping between a state s and a goal g exists and that the goal g exists for every state s. This assumption leads to the definition of an achieved goal g^h to every state s. The HER replaces the original goal g of an episode with the actual result of the episode s, making the action in the episode seems to be intended for the actual result. As a consequence, when the original goal g is substituted with the achieved goal g^h, an unsuccessful experience can be taken as a successful experience and sampling efficiency increases. An experience after the substitution is defined as hindsight experience $(s_t, g^h, a_t, r_t^h, s_{t+1})$, where the reward r_t^h is recomputed for the hindsight goal g^h.

III. PROPOSED METHOD

A. CONCEPT OF PROPOSED METHOD EXPLAINED WITH EXAMPLES

In this section, a deep learning based method to determine the key-action during an episode is proposed. Proposed method consisting of rewards prediction model, key-action detection, and credit assignment stages is incorporated into the off-policy algorithm which is the first and fifth stages of the process flow in Fig.1. Based on the input (s_t, g, a_t) at a timestep t, the rewards prediction model predicts future rewards at subsequent timesteps and detects the actions at specific timesteps that give decisive clues for the future reward values. The CA is conducted for these detected actions. The proposed method combined with an off-policy RL algorithm consists of five stages as depicted in Fig. 1. Explanations on entries of nomenclature are in subsection III-B.

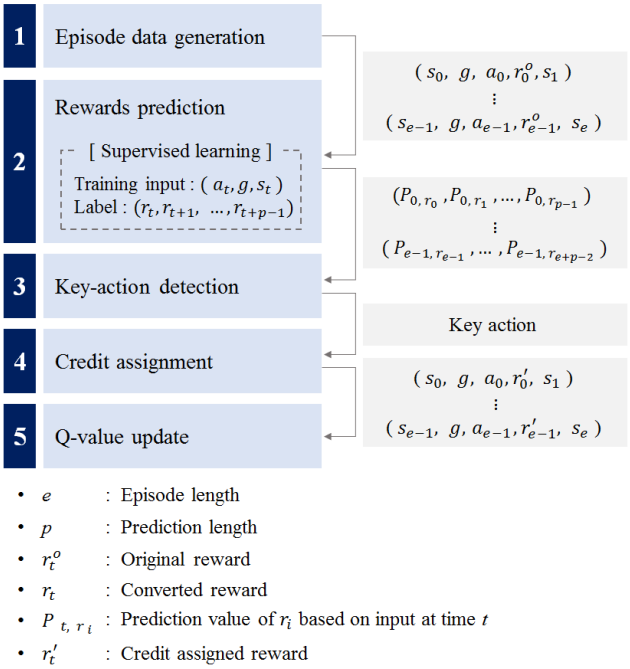


FIGURE 1. Process flow of the proposed method.

Figure 2 represents four sequential episodes in a maze environment and Q value propagation during these four episodes. On the left side of the figure, the shape of the maze and symbol legend are shown. The maze is a 4x4 grid world, and the black blocks are walls that the agent cannot enter. The point marked with S is the start point of the maze, and the point marked with E is the endpoint. The yellow line is the path of the agent for each episode. The darker green block is the location where the action contributing most to escape the maze is detected. Solving a maze is an EDAR task under sparse binary rewards. Thus, the CA_1 strategy is used to choose adjacent-key-actions. Details on CA_1 strategy are in subsection III-B.3. Only when the maze is solved, an episode ends and a reward for success is given. The first row represents the results of each episode and the location where the key-action of that episode is detected. The second and third rows show the propagation of Q value according to the results of the episodes in the first row. The second row represented by 4 mazes as conventional method is based on a general off-policy algorithm, and the third row formed by another 4 mazes as proposed method is based on the key-action detection. The agent moves either right, left, up, or down in each block at each timestep. In each block, one action whose Q value is the largest among the four actions' Q values is indicated by an arrow, and the red color depth of the block varies according to the magnitude of the indicated action's Q value. In the figure, the red color shades darker as the magnitude of Q value becomes larger.

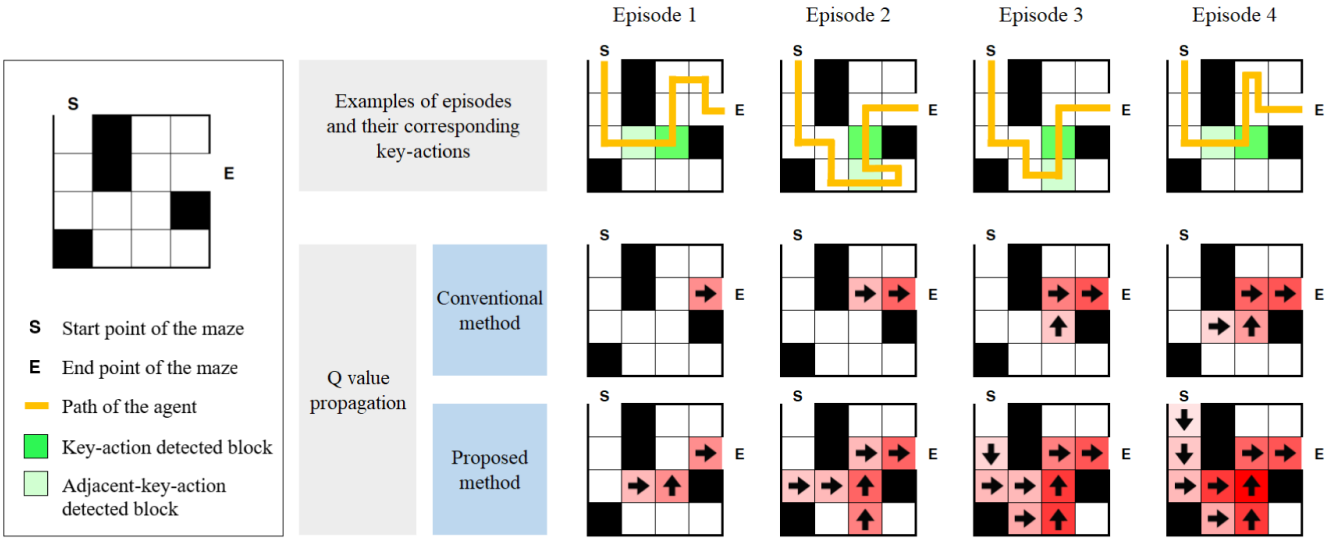


FIGURE 2. An example of EDAR task. In this GridWorld configuration, the agent moves from the start point to the endpoint of the maze to receive a reward. With the proposed method a key-action and adjacent-key-actions are detected and rewards are re-assigned.

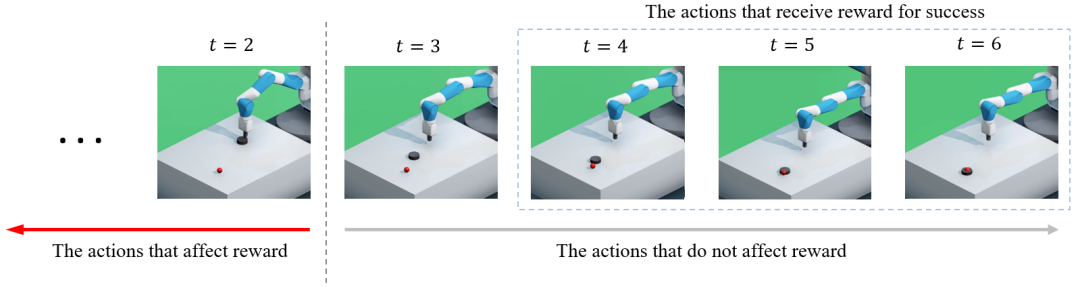


FIGURE 3. An example of non-EDAR task. In the Fetch slide task, the robot actions after hitting the puck do not affect the final reward.

Though the block position where the key-action is detected is identical with 4 episodes, timestep of key-action for each episode is different. The detailed description on key-action found by the proposed method is in subsection III-B. In each episode, the action that goes up from the darker green block to upper block is detected as the key-action. The lighter green block is the location where the adjacent-key-action is detected. In this example, the adjacent-key-action obtained by the CA_1 strategy is taken as the one just before the key-action and the location of each adjacent-key-action depends on the path in each episode.

As the agent succeeded in escaping the maze in the first episode before the terminal state, as in the case of general off-policy algorithm, a reward for success received at the end of the episode is incorporated into the Q value of the action that is taken at the last timestep. In the proposed method, the key-action, as well as the last action of the episode, receives a reward for success, so that the color of the block accounting for the Q value of the key-action also changes to light red. The block with adjacent-key-action is marked in red lighter than that of key-action block because the adjacent-key-action receives a smaller reward than the key-action. As the number of episodes increases sequentially, Q value propagation is performed faster with the proposed method, due to larger number of actions,including adjacent-key-action, that receive rewards. As a result, policy to solve the maze is obtained faster due to the presence of the adjacent-key-action.

Figure 3 is an illustration of the far-zone Fetch slide task at each timestep. Because the far-zone Fetch slide task in this example is a non-EDAR task, the CA_2 strategy is adopted. The agent learns how to make the puck slide to the red goal by hit. An agent hits the puck at timestep $t = 2$, and the puck stopped at the goal at timestep $t = 6$. A reward for success is given when the puck is located within a circle of 5 cm radius centered at the target point. Because the puck is within the circle since timestep $t = 4$, as seen in Fig. 3, the agent gets rewards for success at timesteps $t = 4, 5, 6$. If the proposed method works properly, the key-action to be detected is the action at timestep $t = 2$, namely a_2, and the action a_3 taken between the timestep $t = 2$, when key-action is detected, and timestep $t = 4$, when the agent gets a reward for success for

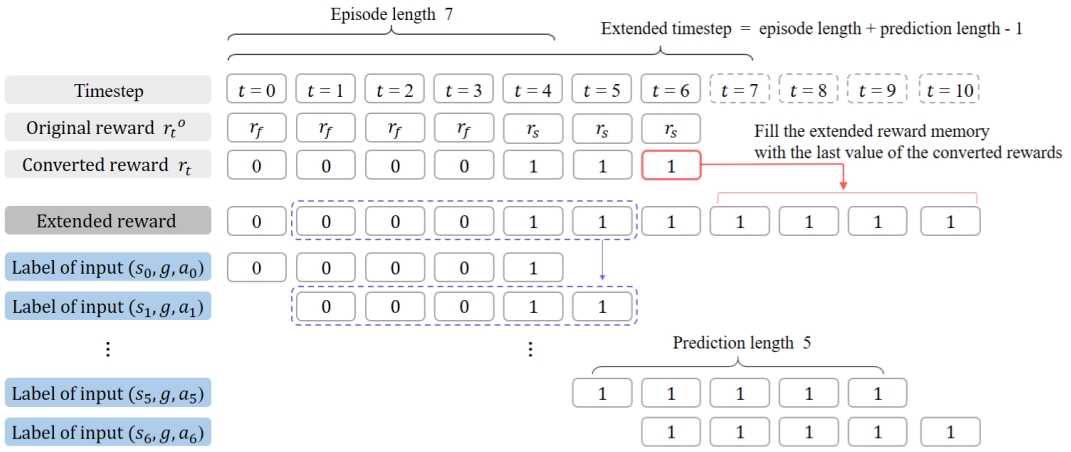


FIGURE 4. Framework for supervised learning of rewards prediction model.

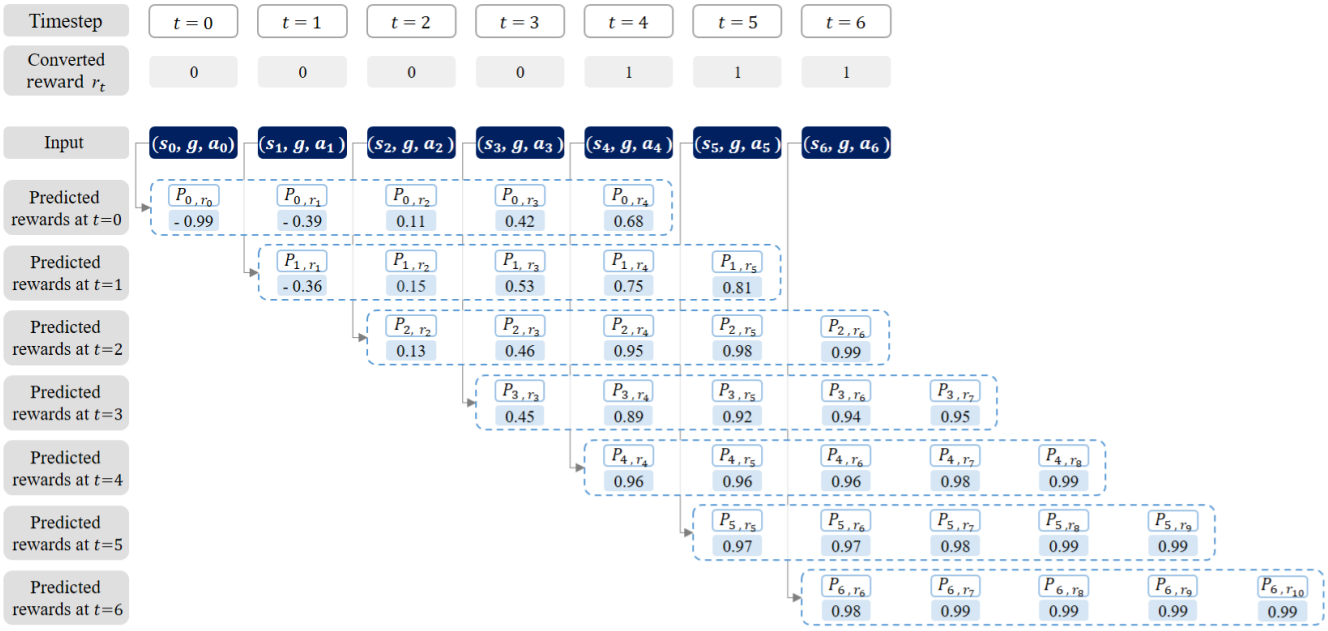


FIGURE 5. Framework for testing rewards prediction model.

the first time after the key-action, is the adjacent-key-action. From timestep $t = 3$, the agent cannot affect the state of the puck because it is beyond the reach of the agent. Giving a reward to the adjacent-key-action a_3 is not for learning the actions, but for enhanced exploration of the actions "hitting the puck" by increasing the Q value of the adjacent-key-actions that cannot be conducted without hitting the puck. This increases the sampling efficiency and the policy achieves the goal faster.

B. PROPOSED ALGORITHM

In this subsection, methodology how to create labels to train the rewards prediction model is explained in Fig. 4. With the trained rewards prediction model, distribution of predicted rewards over timesteps is obtained in Fig. 5. Since finding the input causing abrupt increase of the predicted rewards in a cause and effect relationship is our concern, differential predicted rewards are obtained in Fig. 6 from the distribution of predicted rewards. With differential predicted rewards, a table for key-action detection is made. Based on the entries in the table, the key-action is detected. Following the key-action detection, re-assignment of rewards to key-action and adjacent-key-actions with a relevant formula is executed by CA strategies, as shown in Figs. 7 and 8.

Process flow of the proposed method is shown in Fig. 1. In the first stage of Fig. 1, episodes with each episode length e are generated. Each episode consists of $(s_0, g, a_0, r_0^o, s_1), \ldots, (s_{e-1}, g, a_{e-1}, r_{e-1}^o, s_e)$. With the

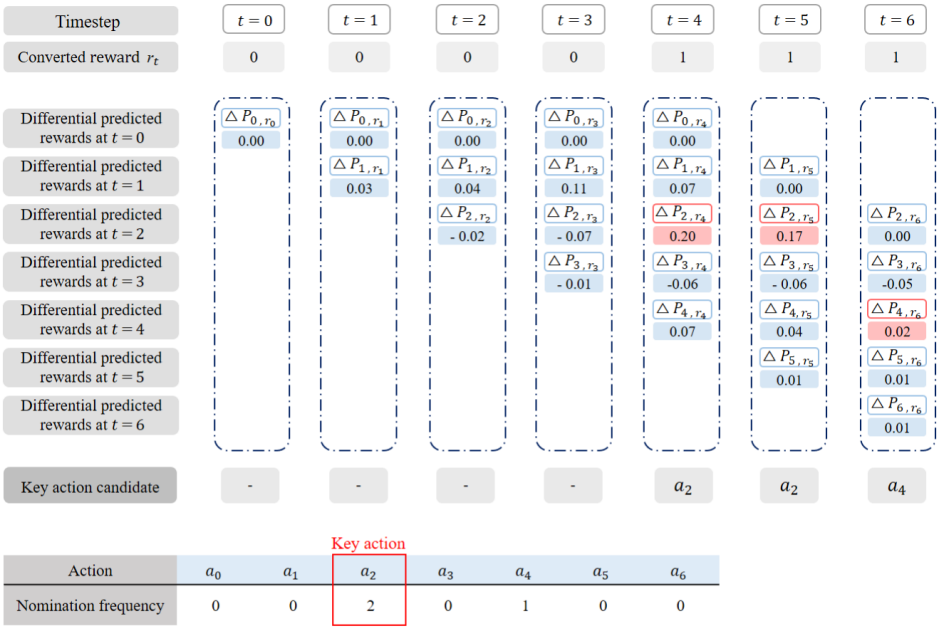
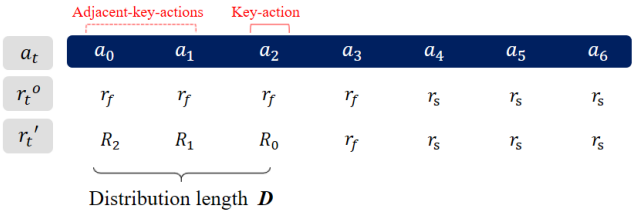
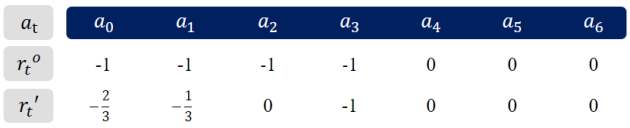


FIGURE 6. Key-action detection process.

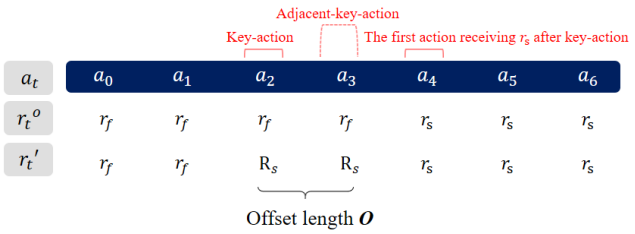


(a) CA_1 strategy with symbols

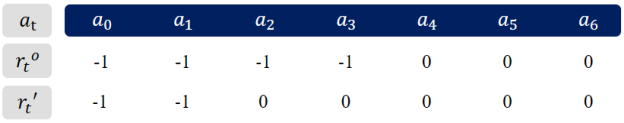


(b) CA_1 strategy with numbers

FIGURE 7. Credit assignment strategy CA_1.



(a) CA_2 strategy with symbols



(b) CA_2 strategy with numbers

FIGURE 8. Credit assignment strategy CA_2.

episodes, each denoted by e 5-tuples, training inputs and labels representing rewards for rewards prediction model are obtained and supervised learning is conducted in the second stage. In this stage, if the original rewards are not $r_t^o = 0, 1$, converted rewards $r_t = 0, 1$ are used as labels for the training. Until the third stage, converted rewards are used. The rewards prediction model predicts the rewards for the subsequent p timesteps, using the prediction length p as a hyperparameter. The P_{t,r_i} is the predicted r_i based on the input (s_t, g, a_t). It is noted here that P_{t,r_i} can take arbitrary value within a range determined by an activation function of the output layer of the rewards prediction model, whereas r_i is binary. Therefore, based on input (s_{e-1}, g, a_{e-1}) of an episode at the last timestep $t = e - 1$, p outputs $P_{e-1,r_{e-1}}$, $P_{e-1,r_e}, \ldots, P_{e-1,r_{e+p-2}}$ are generated, and these values respectively correspond to reward values r_{e-1}, $r_e, \ldots, r_{e+p-2}$. With these predicted rewards, key-actions are detected in the third stage. In the fourth stage, credit assignment is conducted, and recalculated rewards are assigned to the key-actions and their respective adjacent-key-actions. The experiences generated in the first stage are changed to new experiences with credit assigned rewards r'. With these experiences, the policy network is updated in the fifth stage. If the episode length is inadequate, the number of experiences supplied as the input of the second stage can be adjusted properly. The proposed method finds the key-actions within the

experiences given as input. In the same manner, the proposed method can be applied to non-episodic tasks.

Detailed descriptions of the second, third, and fourth stages in Fig. 1 are given with an example. The episode used as the example is the one in Fig. 3. The puck is hit at timestep $t = 2$ and original rewards are -1 and 0. The agent receives rewards for success at timesteps $t = 4, 5, 6$.

1) REWARDS PREDICTION

Figure 4 shows the supervised learning for the rewards prediction model. The first row is the timesteps of the episode illustrated in Fig. 3. Entries in the same column are pertinent to the same timestep. The second row is the original reward r_t^o received at each timestep. In this example, -1 and 0 are used as binary rewards, $r_f = -1$ for failure and $r_s = 0$ for success. The third row presents converted rewards, 0 for failure and 1 for success. The rewards prediction model predicts the future reward values at p consecutive timesteps with their indices starting from the index of input. For example, with prediction length $p = 5$, the predicted reward values based on the input (s_6, g, a_6) are $r_6, r_7, \ldots, r_{10}$. However, because the episode length is 7, rewards $r_7, r_8, \ldots, r_{10}$ do not exist. To deal with this problem, timesteps are extended, and the reward value at the last timestep of the episode is assigned as rewards at the extra timesteps. Thus, the reward value 1 at timestep $t = 6$ is assigned to $r_7, r_8, \ldots, r_{10}$. Extra rewards assigned this way are presented in the fourth row. Labels of input at each timestep are presented from the fifth row to the eleventh row. Note that some rows are absent for illustrational convenience. As an instance, the label (0, 0, 0, 1, 1) for input (s_1, g, a_1) is a 5-tuple consisting of reward values from timestep $t = 1$ to timestep $t = 5$.

Figure 5 shows predicted reward values based on input at each timestep. For example, with input (s_1, g, a_1), rewards prediction model predicts the rewards at timesteps $t = 1, \ldots, 5$ as $-0.36, 0.15, 0.53, 0.75, 0.81$. This means that the probability of obtaining reward value 1 at timestep $t = 1$ and $t = 2$ based on the input at timestep $t = 1$ is relatively low, considering rewards predicted as -0.36, 0.15, and the probabilities of obtaining reward value 1 at timesteps $t = 4$ and $t = 5$ are relatively high. Comparing with the converted rewards 0, 0, 0, 1, 1 at timesteps $t = 1, \ldots, 5$, it can be stated that the rewards prediction model can predict the trend of rewards.

In this example, the agent hits the puck at timestep $t = 2$, so the rewards prediction model is highly likely to assign reward value 1 at timesteps $t = 4, 5, 6$ when it takes (s_2, g, a_2) as input. The predicted rewards $P_{1,r_4} = 0.75$ and $P_{1,r_5} = 0.81$ based on the input at timestep $t = 1$, the time right before the agent hits the puck, are relatively high, but they are not as big as $P_{2,r_4} = 0.95$ and $P_{2,r_5} = 0.98$ obtained from the input at timestep $t = 2$. It is because the agent performs an action at timestep $t = 1$ that is close to the key-action but not decisive as the key-action. The predicted rewards $P_{3,r_4} = 0.89$ and $P_{3,r_5} = 0.92$ based on the input (s_3, g, a_3), the timestep right after the agent hits the puck, are lower than the predicted rewards values obtained at timestep $t = 2$.

2) KEY-ACTION DETECTION

It is seen in Fig. 5 that the distribution of the predicted rewards does not clearly identify the abrupt increase of the predicted rewards. For instance, with the reward r_4, $P_{0,r_4} = 0.68$, $P_{1,r_4} = 0.75$, $P_{2,r_4} = 0.95$, $P_{3,r_4} = 0.89$, $P_{4,r_4} = 0.96$, so the largest predicted reward is P_{4,r_4} and the second largest is P_{2,r_4}, both are closed to each other. Since abrupt change of the predicted rewards according to input over timesteps is our concern, differential predicted rewards in subsequential timesteps are introduced as shown in Fig.6. The differential predicted rewards $\Delta P_{t,r_i}$ can be defined as follows

$$\Delta P_{t,r_i} = \begin{cases} 0, & \text{if } P_{t-1,r_i} \text{ does not exist} \\ P_{t,r_i} - P_{t-1,r_i}, & \text{if } P_{t-1,r_i} \text{ exists} \end{cases} \quad (1)$$

where $\Delta P_{t,r_i}$ is the difference of predicted rewards of r_i obtained at timestep t and at timestep $t-1$. With the aids of the values presented in Fig. 5 and (1), values of $\Delta P_{t,r_i}$ for relevant ranges of t, i with prediction length $p = 5$ are shown in Fig. 6.

Consider differential predicted rewards $\Delta P_{0,r_4} = 0$, $\Delta P_{1,r_4} = 0.07$, $\Delta P_{2,r_4} = 0.20$, $\Delta P_{3,r_4} = -0.06$, $\Delta P_{4,r_4} = 0.07$ in Fig. 6, involving with the reward r_4. Among these differential predicted rewards, the largest one is $\Delta P_{2,r_4} = 0.20$. This means that the key-action that most likely to trigger reward $r_4 = 1$ is a_2. The same process is repeated for r_4, r_5, r_6 whose values are 1. The largest differential predicted reward for each timestep associated with reward value 1 is shown in the boxes filled with light red. The actions at the timestep marked in light red are the key-action candidates of the entire episode. Each key-action candidate that is likely to trigger corresponding reward is represented in tenth row.

The key-action for the CA_1 and CA_2 strategies is defined as an action that is nominated as a key-action candidate most frequently in the episode. The table at the bottom of Fig. 6 shows the frequency of nomination of each key-action candidate. In the table, a_2 is seen to be nominated twice and a_4 is nominated once. Therefore, the action a_2 becomes the key-action. If there are actions that are nominated with the same frequency, the action that is taken at the later timestep is chosen as the key-action, because more actions in episode can take into account the Q value propagated.

Proper prediction length p is necessary for identifying key-action. In the Fetch slide task, a key-action is likely to be the action that is taken at around the moment of hitting the puck. The key-action candidate for r_6 should also be a_2 as well, due to achieved goal since $t = 4$. However, because the rewards prediction model only predicts $r_1, r_2, \ldots, r_5$ when it receives input at timestep $t = 1$, there is no prediction reward P_{1,r_6} and $\Delta P_{2,r_6}$ becomes 0 according to (1). If the prediction length is longer than 5, $\Delta P_{2,r_6}$ might be the biggest one among available differential predicted rewards.

3) CREDIT ASSIGNMENT

Figures. 7 and 8 contain tables showing the CA strategies, CA_1 strategy and CA_2 strategy, which re-assign the reward to the key-action a_2 and adjacent-key-actions in the Fetch slide task depicted in Fig.3. The second row of each table shows the original rewards r_t^o that are given during the episode, and the third row shows the result of CA for the key-action a_2 and adjacent-key-actions.

Figure 7(a) shows the mechanism of the CA_1 strategy. The number of the key-action and adjacent-key-actions is defined as the *distribution length D*. In this example, the distribution length D is set to 3. If D is excessively large, rewards are given to actions that do not affect the reward for success. Thus, it is necessary to choose a proper value for D for efficient propagation of the Q value. Recalculated rewards R_i are assigned by the CA_1 strategy to the key-action and adjacent-key-actions as follows

$$R_i = r_s + \left(\frac{r_f - r_s}{D}\right) i, \tag{2}$$

where i is changed from 0 to (D-1). According to (2), i is the index matched with timestep t. The key-action receives the same reward value of r_s, and adjacent-key-actions receive the reward values between r_f and r_s. Figure 7(b) is an example of the CA_1 strategy when using binary reward values as $r_f = -1$ and $r_s = 0$. The R_0, R_1, and R_2 are calculated by (2). In case of the CA_1 strategy, increment of i corresponds to decreased timestep t.

Figure 8(a) shows the CA_2 strategy. The adjacent-key-actions are selected among the actions between the key-action and another action receiving a reward for success for the first time after the key-action. The number of adjacent-key-actions is defined as the *offset length O*. The offset length O is set to 2, which corresponds to the number of the key-action and adjacent-key-actions, for this example. After applying the CA_2 strategy, the action a_3 becomes the adjacent-key-action and presented in Fig. 8 as R_s like the key-action. Figure 8(b) is an example of the CA_2 strategy when using binary reward values -1 and 0. It can be seen that the reward for action a_3 is changed from -1 to 0.

C. PROPOSED ALGORITHM COMBINED WITH DDPG AND HER

The proposed method improves sampling efficiency and reduces the delay between action and consequent reward by increasing the number of the experiences receiving the reward for success by CA. Because the proposed method conducts CA only for successful episodes, the HER providing successful episodes even out of unsuccessful episodes can be combined to enhance exploration. The pseudo code of the proposed method combined with the HER is presented in Algorithm 1. After a minibatch of episodes is sampled from the replay buffer, the HER is applied and for the successful episodes key-action detection as well as credit assignment based on the CA_1 or CA_2 strategy is performed by the proposed method.

Algorithm 1 Proposed Method: Rewards Prediction Based CA Combined With HER

Given: an off-policy RL algorithm $\mathbb{A}$(DDPG), a strategy $\mathbb{S}$ for sampling goals for replay, a method $\mathbb{K}$ for calculating key-actions, a reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \rightarrow \mathbb{R}$

Initialize neural networks $\mathbb{A}$
Initialize replay buffer R
for epoch = 1, K **do**
  for episode = 1, M **do**
    Sample a goal g and initial state s_0
    for t=0,T-1 **do**
      Sample an action a_t using the behavior policy from $\mathbb{A}$: $a_t \leftarrow \pi(s_t, g) + \mathcal{N}_t$
      Execute the action a_t and observe new state s_{t+1}
      Calculate reward $r_t = r(s_t, g, a_t)$
      Store the experience $(s_t, g, a_t, r_t, s_{t+1})$ in replay buffer R
    end for
    for t=1,N **do**
      Sample a minibatch B from the replay buffer R
      Sample a set of experiences in B with sampling rate ϵ to substitute by achieved goals G sampled with $\mathbb{S}$
      for $g^h \in G$ **do**
        $r_t^h = r(s, g^h, a)$
        Substitute the experience $(s_t, g, a_t, r_t, s_{t+1})$ with the experience $(s_t, g^h, a_t, r_t^h, s_{t+1})$
      end for
      Get the successful episodes E in the minibatch B
      for $e \in E$ **do**
        Calculate the key-action for episode e with $\mathbb{K}$
        Re-assign credit for the key-action and adjacent-key-action experiences $(s_t, g, a_t, r_t, s_{t+1})$ in the minibatch B
      end for
      Perform one step of optimization using $\mathbb{A}$ and minibatch B
    end for
  end for
end for

IV. EXPERIMENT

A. EXPERIMENT ENVIRONMENT

Experiments are conducted with the Fetch environment for multigoal continuous control tasks described in [24]. The Fetch environment is developed with the integration of OpenAI gym [21] and the MuJoCo physics engine [25]. The proposed method is tested with the Fetch push and the Fetch slide tasks. In the Fetch push task, a robot pushes a box to a goal position. In the Fetch slide task, a robot tries to achieve a

goal position by hitting a puck. An episode is successful if the distance between the puck/box and the goal is less than 5 cm.

For the DDPG used as the off-policy RL algorithm, actor and critic networks take a multi-layer perceptron architecture with rectified linear unit (ReLU) as nonlinear activation functions. They are trained by backpropagation using the ADAM [26] as an optimizer. Other values used for training are adopted from [24] and listed in Appendix A. The proposed method uses a multi-layer perceptron with 4 feed-forward layers with 64 neurons with hyperbolic tangent as nonlinear activation functions. The episode length and the prediction length are set to 50, and the nomination frequency of the action needs to be at least two to be selected as a key-action. The rewards prediction model is also trained by backpropagation with ADAM as the optimizer. Number of epochs for training is 100, each consisting of 10^5 episodes, and the batch size is 128 and the learning rate is 10^{-4}. Though it is claimed in the introduction that the CA_1 strategy is more appropriate for the Fetch push task and Fetch slide task with goals in near zone and the CA_2 strategy is more efficient in the far-zone Fetch task, they are compared in the experiments to verify the claim.

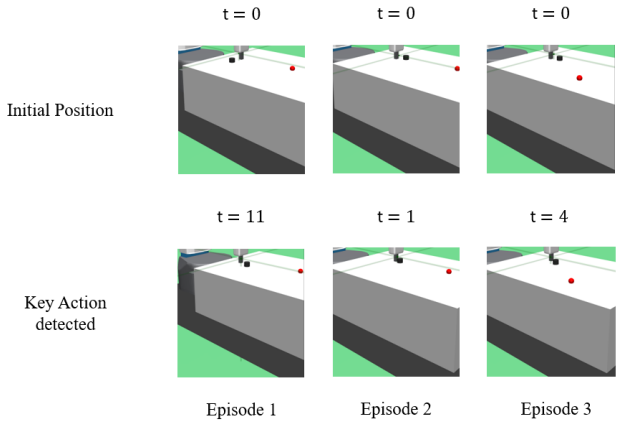


FIGURE 9. Key-actions detected by the proposed method in three episodes of the Fetch slide task. Reward is given only when the puck achieves the goal. Proposed method identifies the key-action when the robot hits the puck in the direction towards the goal. Furthermore, the delay in assigning rewards can be shortened by giving credit directly to the key-action. As an instance, in Episode 1, the gripper is initially located at timestep $t = 0$ to the right of the puck. Proposed method detects the key-action at timestep $t = 11$ when the puck is hit, after the gripper moved behind the puck.

B. RESULTS

In this sebsection, experimental results obtained with the DDPG+HER, CA_1 strategy, and CA_2 strategy are presented. It is to be shown that the CA_1 strategy works better than the others with the Fetch push task and near-zone Fetch slide task, while the CA_2 strategy is more efficient with far-zone Fetch slide task. Therefore, the results suggest use of the CA_1 strategy for the Fetch push and near-zone Fetch slide tasks and the CA_2 strategy for far-zone Fetch slide task. Provided that the CA_1 strategy is used for far-zone Fetch slide task and the CA_2 strategy is used for the Fetch push and near-zone Fetch slide task, their performances are close to those of the DDPG+HER.

1) EXPERIMENT A. VISUAL CONFIRMATION OF THE DETECTED KEY-ACTIONS BY THE PROPOSED METHOD IN A NON-EDAR TASK

The key-action detected by the proposed method is illustrated in Fig. 9. It is seen that the proposed method identifies the moment when the robot gripper hits the puck in a successful episode for the Fetch slide task. Based on the information of the key-action, it is possible to redistribute the reward by giving credit directly to the key-action. This process drastically reduces the delay in assigning the reward and, when properly designed, directs the agent toward good policies.

2) EXPERIMENT B. COMPARISON BETWEEN THE PROPOSED METHOD AND DDPG+HER FOR AN EDAR TASK (FETCH PUSH TASK)

Experimental results obtained from the proposed method are presented together with the results of DDPG+HER for the Fetch push task. For the HER, the rate of hindsight experience in sampling a minibatch is kept at 0.8 and the *final* strategy [20], where the goal reached in the last timestep of the episode is chosen as the achieved goal, is used for sampling goals. An extensive search is performed and the distribution length is set to 5. The offset length is set to the number of all actions, between the key-action and the action receiving the reward for success for the fist time after the key-action, plus 1, representing inclusion of the key-action. The training is performed for 200 epochs. For each training epoch, 50 cycles are executed with two agents in the experiment environment and stored in the replay buffer, followed by 40 optimization steps of the policy. After each training epoch, 20 test episodes are executed and the success rate is calculated. To reduce granularity at each epoch index, moving average of the success rate across 50 epochs is taken. The foregoing process is repeated with multiple random seeds. For each epoch index, the median is selected and presented as the result. The same experimental setup is used in Experiment C and Experiment D.

Comparison of the DDPG+HER with and without the proposed method is presented in Fig. 10. The results show that the proposed method improves the convergence speed and achieves similar performance to that of DDPG+HER over 200 epochs. The success rate in the last epoch of CA_1 strategy, CA_2 strategy, DDPG+HER are 100%, 98%, 98%, respectively. It is notable that the CA_2 strategy not considering actions after the key-action is seen to be able to solve the Fetch push task with comparable convergence speed as compared to DDPG+HER.

Variation of mean Q value obtained during training with DDPG+HER and proposed method is shown in Fig. 11. As higher mean Q value of proposed method is indicative of more aggressive policy, the goal is achieved faster. More aggressive policy is more likely to explore various parts of

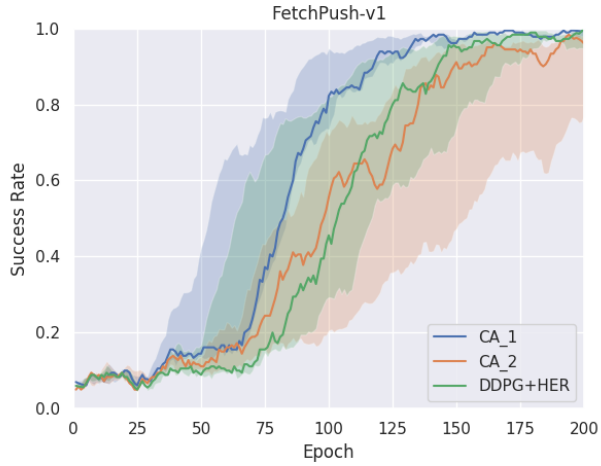


FIGURE 10. Success rate obtained by proposed method and DDPG+HER for the Fetch push task. Results show that proposed method achieves similar performance to that of DDPG+HER over 200 epochs. Success rates of CA_1 strategy, CA_2 strategy, DDPG+HER are 100%, 98%, 98%, respectively. It is seen that CA_1 strategy converges faster as compared to others.

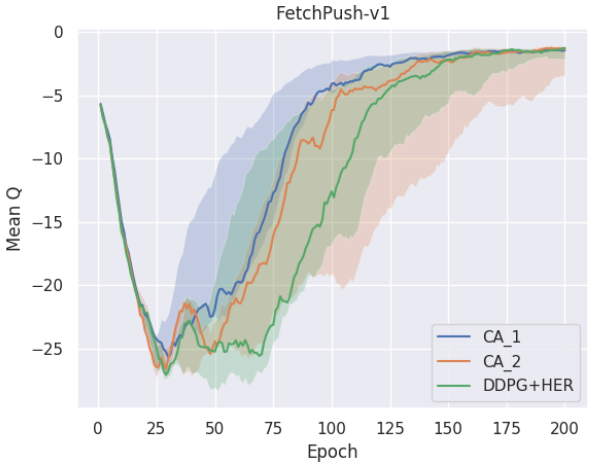


FIGURE 11. Variation of mean Q value during training with DDPG+HER and proposed method (CA_1 and CA_2 strategies) for the Fetch push task. At epoch index 200, the mean Q value is approximately −2 with DDPG+HER and proposed method. It is seen that CA_1 and CA_2 strategies converge to the optimum mean Q value −2 faster, which means more efficient exploration during training.

the state space, leading to enhanced sampling efficiency. At epoch index 200, the mean Q value is approximately −2 with DDPG+HER and proposed method. It is seen that the CA_1 and CA_2 strategies converge to the optimum mean Q value −2 faster as compared to DDPG+HER.

The performance is measured by the quality of the best policy over 1000 test episodes. The average success rate is shown in TABLE 1. The CA_1 strategy performs better than DDPG+HER and CA_2 strategy. The success rate of CA_1 strategy is 99.17% on average with the test episodes, which exceeds 97.67% of DDPG+HER and 88.83% of CA_2 strategy.

TABLE 1. Comparison between the proposed method and DDPG+HER for the Fetch push task.

Algorithm	Push succ %
DDPG+HER	97.67% ± 1.5%
CA_1 strategy	**99.17% ± 0.15%**
CA_2 strategy	88.83% ± 7.25%

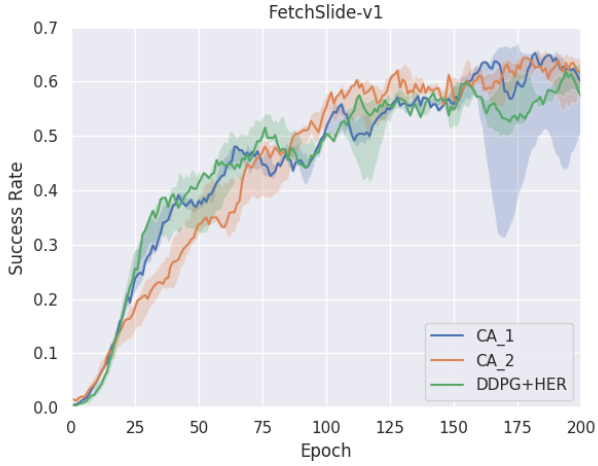


FIGURE 12. Success rate obtained by proposed method and DDPG+HER for Fetch slide task. Results show that the proposed method achieves similar performance to that of DDPG+HER over 200 epochs. Success rates of CA_2 strategy, CA_1 strategy, DDPG+HER are 62%, 60%, 57%, respectively. This success rate accounts for the Fetch slide task in near zone as well, which conceals higher success rate in far zone with the CA_2 strategy. The higher success rate in far zone is referred to Fig. 14.

3) EXPERIMENT C. COMPARISON BETWEEN PROPOSED METHOD AND DDPG+HER FOR A NON-EDAR TASK (FETCH SLIDE TASK)

Fig. 12 compares DDPG+HER with and without the proposed method for Fetch slide task. The results show that the proposed method achieves similar convergence speed to that of DDPG+HER over 200 epochs. The approximate success rate in the last epoch of CA_2 strategy, CA_1 strategy, DDPG+HER are 62%, 60%, 57%, respectively. It is important to note that, despite the similar convergence speed between epoch index 150 and epoch index 200, the strategies CA_1 and CA_2 strategies are able to achieve higher success rates when compared to DDPG+HER. Because higher success rate is translated into improved policy, the CA_1 and CA_2 strategies demonstrating higher success rates in relation to DDPG+HER, as shown in TABLE 2, are likely to provide improved policies.

Variation of mean Q value obtained during training is presented in Fig. 13. It is seen by the higher Q values that the CA_2 strategy is the most aggressive policy. The CA_2 strategy tends to focus during the training on the first "hit" of the puck, which leads to a better exploration of the goals on the table surface. The higher Q value, the faster the policy is able to achieve the goals. At epoch index 200, mean Q values are

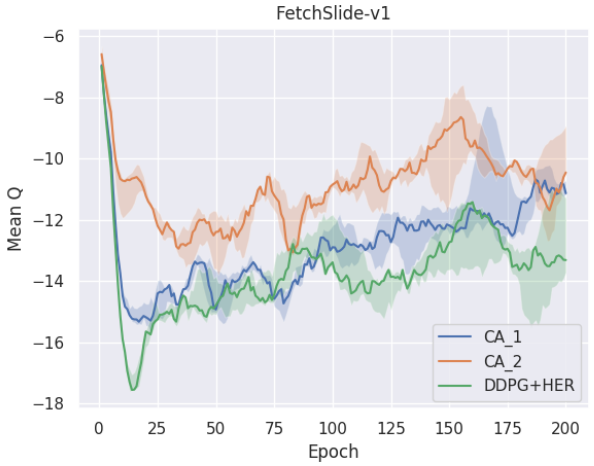


FIGURE 13. Variation of mean Q value during training with DDPG+HER and the proposed method for Fetch slide task. At epoch index 200, the mean Q value is approximately −11, −10.5, and −13.3 with the CA_1 strategy, CA_2 strategy and DDPG+HER respectively. It is seen that CA_2 strategy takes higher Q values as compared to CA_1 strategy and DDPG+HER, which means more efficient exploration of the state space during training.

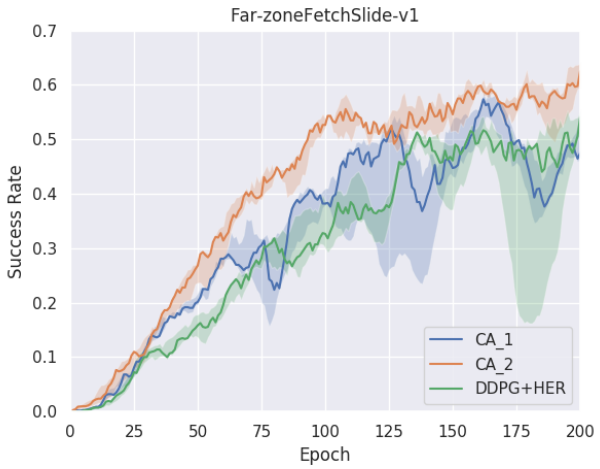


FIGURE 14. Success rate obtained by proposed method and DDPG+HER for the far-zone Fetch slide task. Success rates of CA_2 strategy, CA_1 strategy, DDPG+HER are 62%, 48%, 49%, respectively. It is seen that CA_2 strategy outperforms CA_1 strategy and DDPG+HER by 10 percent at epoch index 200.

approximately −11, −10.5, and −13.3 with the CA_1 strategy, CA_2 strategy, and DDPG+HER respectively.

TABLE 2. Comparison between the proposed method and DDPG+HER for the Fetch slide task.

Algorithm	Slide succ %
DDPG+HER	61.37% ± 1.29%
CA_1 strategy	**68.77% ± 1.34%**
CA_2 strategy	65.57% ± 0.90%

TABLE 3. Comparison between the proposed method and DDPG+HER for the far-zone Fetch slide task.

Algorithm	Far-zone slide succ %
DDPG+HER	60.45% ± 0.54%
CA_1 strategy	58.63% ± 2.69%
CA_2 strategy	**67.38% ± 0.38%**

The performance in the Fetch slide task is measured by quality of the best policy over 1000 test episodes. The average success rate is shown in TABLE 2. As shown in the table, CA_1 strategy performs better than DDPG+HER and CA_2 strategy. The success rate of CA_1 strategy is 68.77% on average with the test episodes, which exceeds 65.57% of CA_2 strategy and 61.37% of DDPG+HER.

4) EXPERIMENT D. COMPARISON BETWEEN THE PROPOSED METHOD AND DDPG+HER FOR A NON-EDAR TASK (FAR-ZONE FETCH SLIDE TASK)

Comparison of the DDPG+HER with and without the proposed method for the far-zone Fetch slide task is presented in Fig. 14. As goals reachable by the robot are excluded,

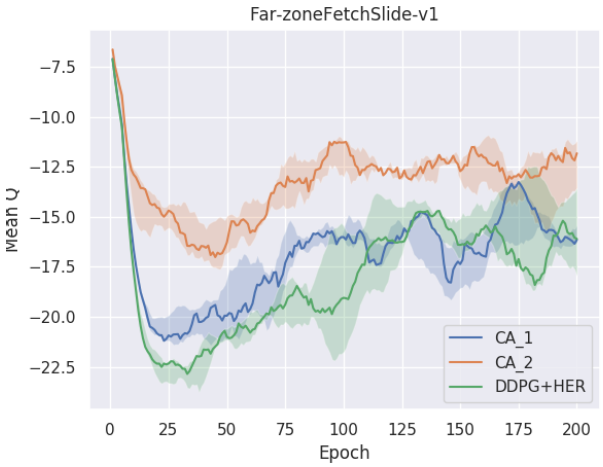
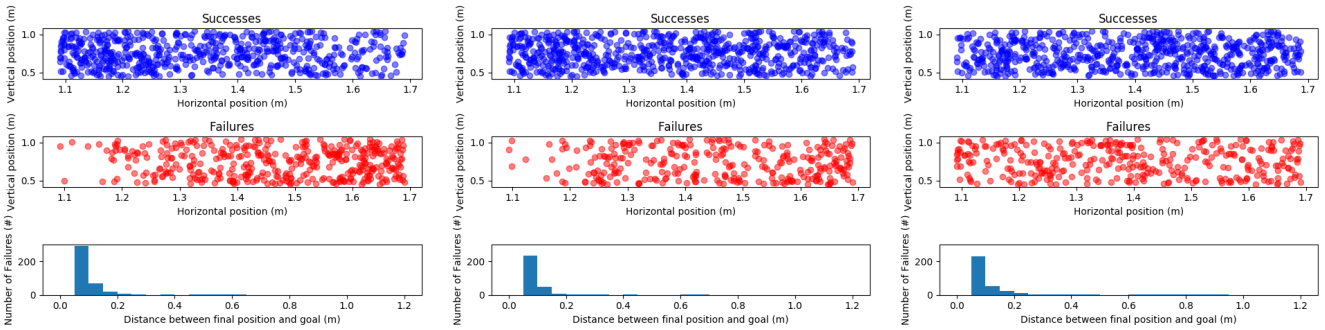


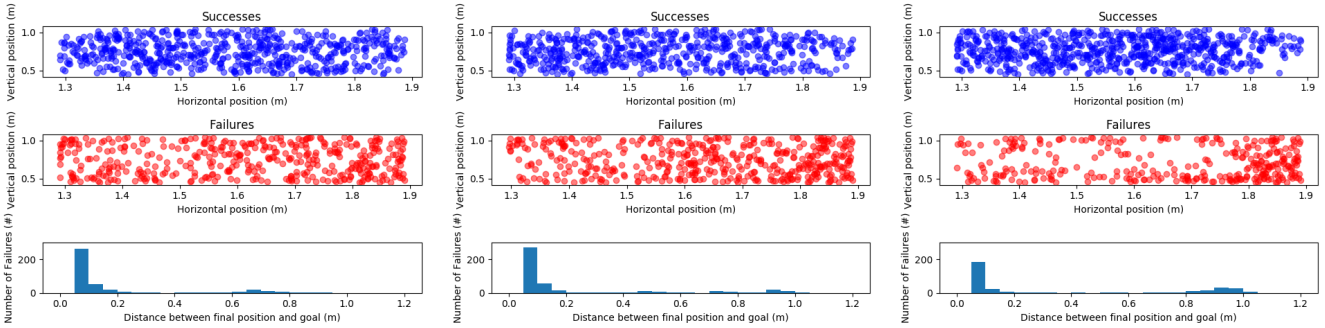
FIGURE 15. Variation of mean Q value during training with DDPG+HER and the proposed method for the far-zone Fetch slide task. At epoch index 200, the mean Q value is approximately −12.3, −16.1, and −16.2 for the CA_2 strategy, DDPG+HER, and CA_1 strategy respectively. It is seen that CA_2 strategy takes higher Q values as compared to CA_1 strategy and DDPG+HER, which means that the policy trained with CA_2 strategy is able to achieve the goal faster and explore goals located in far zone.

the difficulty level of the task is increased. The robot can not push the puck until the goal position as compared to the Fetch slide task. Proposed method, more specifically the CA_2 strategy, increases the convergence speed and success rate when compared with DDPG+HER over 200 epochs. It is observed in Fig. 14 that the variance of success rate at each epoch index is decreased in general over entire epochs with the CA_2 strategy as compared to that of CA_1 strategy and DDPG+HER. The final success rate at epoch index 200 of CA_2 strategy, CA_1 strategy, DDPG+HER are 62%, 48%, 49%, respectively. Similar to the results obtained in Experiment C, the proposed method is able to achieve



(a) DDPG+HER (b) CA_1 strategy (c) CA_2 strategy

FIGURE 16. Dot maps of success and failure obtained by the best policy over 1000 test episodes, gained with the DDPG+HER and the proposed method for the Fetch slide task. Each column consists of two dot maps, each representing success dot map (uppermost one) and failure dot map(middle one), and histogram(lowermost one) of distance between the final position of the puck and the goal for each episode. The DDPG+HER and the CA_1 strategy are able to achieve more goals in the near zone between located within 1.2m of the horizontal position. The CA_2 strategy, on the other hand, achieves more goals located in the far zone beyond 1.2m in the horizontal position. Higher density of dots in local zone represents higher chance of event(success or failure) occurrence. See the variation of local density of dots along horizontal position. (a) DDPG+HER. (b) CA_1 strategy. (c) CA_2 strategy.



(a) DDPG+HER (b) CA_1 strategy (c) CA_2 strategy

FIGURE 17. Dot maps of success and failure obtained by the best policy over 1000 test episodes, gained with the DDPG+HER and the proposed method for the Fetch slide task. Each column consists of two dot maps, each representing success dot map (uppermost one) and failure dot map(middle one), and histogram(lowermost one) of distance between the final position of the puck and the goal for each episode. The CA_2 strategy is highly accurate and clearly outperforms the CA_1 strategy and the DDPG+HER with goals located in the zone between 1.5m and 1.8m of the horizontal position. (a) DDPG+HER. (b) CA_1 strategy. (c) CA_2 strategy.

higher success rate when compared with DDPG+HER during training.

Variation of mean Q value is shown in Fig. 15. As the goals are located far from the robot, the propagation of rewards combined with exploration in distant locations on the table is a critical problem. Similar to the pattern observed in Experiment C, in the far-zone Fetch slide task the CA_2 strategy also shows higher mean Q value as compared to other techniques. At epoch index 200, mean Q values are approximately -12.3, -16.2, and -16.1 with the CA_2 strategy, CA_1 strategy, and DDPG+HER respectively. The mean Q values obtained in Fig. 15 are lower as compared to the ones obtained in Fig. 13 because the goals are located farther from the robot.

The performance is measured by evaluating the quality of the best policy over 1000 test episodes. The average success rate is shown in TABLE 3. The CA_2 strategy is successful on average 67.38% of the test episodes, which exceeds 58.63% of CA_1 strategy and 60.45% of DDPG+HER. The results show that the reduction of the delay of the CA_2 strategy between the key-action and assigning reward improves the final success rate in the far-zone Fetch slide task. Because the far-zone Fetch slide task is more difficult to execute, lower success rate with the far-zone Fetch slide task is expected in the experiments, however, the results shown in TABLE 2 and another results listed in TABLE 3 are comparable as opposed to the expectation.

5) EXPERIMENT E. ANALYSIS OF SUCCESS AND FAILURE OBTAINED FROM THE BEST POLICY FOR THE FETCH SLIDE AND FAR-ZONE FETCH SLIDE TASKS

It is important to see by an analysis which goals are achievable by the policy and which ones are not, because such characterization enables efficient use of the policy. To perform the analysis, dot maps of success and failure obtained by the best policy over 1000 test episodes can be used. Figs. 16, 17 show dot maps gained with the DDPG+HER and proposed method for the Fetch slide and far-zone Fetch slide tasks. Each column of Figs. 16,17 consists of two dot maps, each representing success dot map (uppermost one) and failure dot map(middle one), and histogram(lowermost

one) of distance between the final position of the puck and the goal for each episode. Horizontal axis of dot map represents horizontal position and vertical axis indicates vertical position on the table. Blue dots presented in success dot map represent achieved goals and red dots in failure dot map indicate unachieved goals. The sum of the number of dots in the success dot map and the number of dots in the failure dot map is the number of test episodes. The histogram of distance shows the distribution of relative frequency of distance between the final position of the puck and the goal for each episode that ended in failure.

For the Fetch slide task in Fig. 16 with goals located within 1.2 m, the robot is able to reach and thus can push the puck to the goal. The results in Experiment C shows that the CA_1 strategy is the best one for this task. It is seen that both DDPG+HER and CA_1 strategy are able to learn how to push the puck to the goal. The CA_2 strategy, however, just learns to hit the puck to achieve the goal, even if it is in a reachable position. Therefore, the success rate of the CA_2 strategy with the goals in near zone is worse than CA_1 strategy and DDPG+HER. For goals in far zone, the success rates of the CA_1 strategy and DDPG+HER are lower than that of the CA_2 strategy.

It is seen in Fig. 17 that the horizontal location of the goal of far-zone Fetch slide task is between 1.3 m and 1.9 m. The results in Experiment D show that the CA_2 strategy is the best one for this task. It is possible to see that CA_2 strategy clearly outperforms CA_1 strategy and DDPG+HER from 1.5m to 1.8m, having a high accuracy for this range.

In Figs. 16 and 17, the histograms show the distance between the final position of the puck and the goal for the episodes that ended in failure. In the experiments, to define a successful episode, a tolerance of 5 cm between the final puck position and goal is used as described in [24]. The results show that the distances between the final puck position and the goal of the majority of the episodes that ended in failure are less than 0.2m with the three approaches. Frequency of failure with the CA_1 strategy and CA_2 strategy is lower than that with the DDPG+HER. The results show that the best policy learned by the proposed method is more accurate.

V. CONCLUSION

This paper deals with the problem in high-dimensional RL environment under sparse binary rewards. In order to increase success rate and convergence speed during training in this environment, a novel method composed of key-action detection and CA strategies is proposed. Rewards prediction model is established and with the rewards predicted by the model, the key-action is identified. Using CA strategies, adjacent-key-actions are determined and then rewards are re-assigned to the key-action and adjacent-key-actions. The CA strategies considered in this paper are CA_1 strategy, which selects adjacent-key-actions among the series of actions before the key-action, and CA_2 strategy, which chooses adjacent-key-actions between the key-action and an action receiving a reward for success for the first time after the key-action.

The CA_1 strategy is fit to the type of EDAR tasks. Every action in EDAR task affects the value of the reward. The CA_2 strategy is efficient to the type of non-EDAR tasks. In non-EDAR tasks, some actions affect the value of the reward. The Fetch push task, where the goal is achieved more effectively by push action, can be classified into EDAR tasks, whereas the Fetch slide task with goals positioned in far zone, requiring hit action to achieve goals, can be classified into non-EDAR tasks. The Fetch slide task with goals positioned in far zone is particularly called in this paper far-zone Fetch slide task. When the goals are located in near zone, the Fetch slide task is better fulfilled by push action. Therefore, the Fetch slide task with goals located in near zone can be classified into EDAR tasks. Entire goal space is divided into near zone, where the goal is reachable by robotic arm, and far zone. Though the key-action with non-EDAR task is relatively easy to identify, the key-action with EDAR task is significantly more difficult to identify. The detected key-actions with certain type of EDAR task are often against our conjecture.

In order to verify the performance of the proposed method, five experiments are conducted. In the experiments, the proposed method demonstrates increase success rate and convergence speed. More specifically, for the Fetch push task, the CA_1 strategy outperforms DDPG+HER, especially increasing the convergence speed. For the Fetch slide task, the CA_1 and CA_2 strategies achieve similar convergence speed but significantly increase the success rate to 68.77% and 65.57% respectively compared to 61.37% obtained by existing method DDPG+HER. For the far-zone Fetch slide task, the CA_2 strategy increases the convergence speed and improves the success rate to 67.38% when compared to 60.45% of DDPG+HER and 58.63% by CA_1 strategy. From the experiments, a guideline for selecting CA strategy according to goal location is provided through goal distribution analysis with dot map. As a result, the CA_1 strategy works in near zone better than CA_2 strategy and DDPG+HER, which corresponds to the Fetch push task and Fetch slide task with goals positioned in near zone, whereas the CA_2 strategy performs in far zone better than CA_1 strategy and DDPG+HER. Therefore, combined use of CA_1 and CA_2 strategies is recommended.

APPENDIX A
NETWORK ARCHITECTURE AND HYPERPARAMETERS

The hyperparameters used for experiments are adopted from [24] describing experiment environment of the HER. List of the hyperparameters used for experiments is as follows

- Actor and critic networks: 3 layers with 256 units each and ReLU non-linearities after each layer
- ADAM optimizer [26] with learning rate 10^{-3} for training both actor and critic networks
- Buffer size: 10^6 experiences
- Polyak-averaging coefficient: 0.95
- Action L2 norm coefficient: 1.0
- Observation clipping: $[-200, 200]$

- Batch size: 256
- Rollouts per MPI worker: 2
- Number of MPI workers: 1 (Fetch push) and 4 (Fetch slide and far-zone Fetch slide)
- Cycles per epoch: 50
- Batches per cycle: 40
- Test rollouts per epoch: 10
- Probability of random actions: 0.3
- Scale of additive Gaussian (exploration) noise: 0.2
- Normalized clipping: $[-5, 5]$

Details of the hyperparameters are described in [20] and [24]. An episode is defined as 50 environment timesteps. One environment timestep is consisted of 20 MuJoCo (simulator) steps with $\Delta t = 0.002s$. The inputs of the neural networks are normalized to have zero mean and unit standard deviation. Normalized clipping is also used.

REFERENCES

[1] 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, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[2] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. 33rd Int. Conf. Mach. Learn.*, vol. 48, Jun. 2016, pp. 1928–1937.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[4] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," Dec. 2017, *arXiv:1712.01815*. [Online]. Available: https://arxiv.org/abs/1712.01815

[5] Y. Wen, J. Si, A. Brandt, X. Gao, and H. Huang, "Online reinforcement learning control for the personalization of a robotic knee prosthesis," *IEEE Trans. Cybern.*, to be published.

[6] A. E. L. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electron. Imag.*, vol. 19, pp. 70–76, Jan. 2017.

[7] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, "Learning invariant feature spaces to transfer skills with reinforcement learning," Mar. 2017, *arXiv:1703.02949*. [Online]. Available: https://arxiv.org/abs/1703.02949

[8] R. S. Sutton, "Temporal credit assignment in reinforcement learning," Ph.D. dissertation, Dept. Comput. Inf. Sci., Univ. Massachusetts, Amherst, MA, USA, 1984.

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

[10] F. S. He, Y. Liu, A. G. Schwing, and J. Peng, "Learning to play in a day: Faster deep reinforcement learning by optimality tightening," Nov. 2016, *arXiv:1611.01606*. [Online]. Available: https://arxiv.org/abs/1611.01606

[11] A. Laud and G. DeJong, "The influence of reward on the speed of reinforcement learning: An analysis of shaping," in *Proc. 33rd Int. Conf. Mach. Learn. (ICML)*, 2003, pp. 1–8.

[12] J. Xie, Z. Shao, Y. Li, Y. Guan, and J. Tan, "Deep reinforcement learning with optimized reward functions for robotic trajectory planning," *IEEE Access*, vol. 7, pp. 105669–105679, 2019.

[13] A. Laud and G. DeJong, "Reinforcement learning and shaping: Encouraging intended behaviors," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2002, pp. 355–362.

[14] C. Lopez, J. R. Marti, and S. Sarkaria, "Distributed reinforcement learning in emergency response simulation," *IEEE Access*, vol. 6, pp. 67261–67276, 2018.

[15] O. Marom and B. Rosman, "Belief reward shaping in reinforcement learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–8.

[16] B. Marthi, "Automatic shaping and decomposition of reward functions," in *Proc. 24th Int. Conf. Mach. Learn. (ICML)*, 2007, pp. 601–608.

[17] H. Van Seijen, M. Fatemi, J. Romoff, R. Laroche, T. Barnes, and J. Tsang, "Hybrid reward architecture for reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 5396–5406.

[18] K. Narasimhan, T. Kulkarni, and R. Barzilay, "Language understanding for text-based games using deep reinforcement learning," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Lisbon, Portugal, 2015, pp. 1–11.

[19] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Puerto Rico, 2016, pp. 1–21.

[20] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 1–11.

[21] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI gym," Jun. 2016, *arXiv:1606.01540*. [Online]. Available: https://arxiv.org/abs/1606.01540

[22] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2018, pp. 6292–6299.

[23] 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," Sep. 2015, *arXiv:1509.02971*. [Online]. Available: https://arxiv.org/abs/1509.02971

[24] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, V. Kumar, and W. Zaremba, "Multi-goal reinforcement learning: Challenging robotics environments and request for research," Feb. 2018, *arXiv:1802.09464*. [Online]. Available: https://arxiv.org/abs/1802.09464

[25] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2012, pp. 5026–5033.

[26] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–15.

MINAH SEO received the B.Sc. degree in mechanical engineering and logistics from Inha University, South Korea, in 2017. She is currently pursuing the M.S. degree with the Cho Chun Shik Graduate School of Green Transportation, Korea Advanced Institute of Science and Technology (KAIST). Her research interests include machine learning and data processing.

LUIZ FELIPE VECCHIETTI received the B.Sc. degree in electronics and computer engineering and the M.Sc. degree from the Federal University of Rio de Janeiro, in 2015 and 2017, respectively. He is currently pursuing the Ph.D. degree with the Cho Chun Shik Graduate School of Green Transportation, Korea Advanced Institute of Science and Technology (KAIST). His research interests include machine learning, digital signal processing, and applied deep learning.

SANGKEUM LEE received the bachelor's degree in electrical engineering from Korea University, in 2016, and the M.S. degree from KAIST, where he is currently pursuing the Ph.D. degree. He is also a member of the vehicular Intelligence Laboratory, Cho Chun Shik Graduate School for Green Transportation, KAIST. His current research interests include sensor network systems and its supporting technologies, such as hybrid electric vehicles, electric powered airplanes, sensor networks, and power systems.

DONGSOO HAR received the B.Sc. and M.Sc. degrees in electronics engineering from Seoul National University, and the Ph.D. degree in electrical engineering from Polytechnic University, Brooklyn, NY, USA. He is currently a Faculty Member with KAIST. He authored and published more than 100 articles in international journals and conferences. His current research interests include the optimization of communication system operation and transportation system development with embedded artificial intelligence. He was the member of advisory board, a program chair, a vice chair, and a general chair of international conferences. He was a recipient of the Best Paper Award (Jack Neubauer Award) from the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, in 2000. He is an Associate Editor of the IEEE SENSORS JOURNAL. He also presented invited talks and keynote in international conferences.