

Episodic Self-Imitation Learning with Hindsight

Tianhong Dai ^{1,*}, Hengyan Liu ² and Anil Anthony Bharath ¹

¹ Department of Bioengineering, Imperial College London, London SW7 2AZ, UK; a.bharath@imperial.ac.uk

² Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2AZ, UK; hengyan.liu15@imperial.ac.uk

* Correspondence: tianhong.dai15@imperial.ac.uk

Received: 12 September 2020; Accepted: 16 October 2020; Published: 21 October 2020



Abstract: Episodic self-imitation learning, a novel self-imitation algorithm with a trajectory selection module and an adaptive loss function, is proposed to speed up reinforcement learning. Compared to the original self-imitation learning algorithm, which samples good state–action pairs from the experience replay buffer, our agent leverages entire episodes with hindsight to aid self-imitation learning. A selection module is introduced to filter uninformative samples from each episode of the update. The proposed method overcomes the limitations of the standard self-imitation learning algorithm, a transitions-based method which performs poorly in handling continuous control environments with sparse rewards. From the experiments, episodic self-imitation learning is shown to perform better than baseline on-policy algorithms, achieving comparable performance to state-of-the-art off-policy algorithms in several simulated robot control tasks. The trajectory selection module is shown to prevent the agent learning undesirable hindsight experiences. With the capability of solving sparse reward problems in continuous control settings, episodic self-imitation learning has the potential to be applied to real-world problems that have continuous action spaces, such as robot guidance and manipulation.

Keywords: deep reinforcement learning; hindsight experience replay; imitation learning; exploration

1. Introduction

Reinforcement learning (RL) has been shown to be very effective in training agents within gaming environments [1,2], particularly when combined with deep neural networks [2–4]. In most tasks settings that are solved by RL algorithms, reward shaping is an essential requirement for guiding the learning of the agent. Reward shaping, however, often requires significant quantities of domain knowledge that are highly task-specific [5] and, even with careful design, can lead to undesired policies. Moreover, for complex robotic manipulation tasks, manually designing reward shaping functions to guide the learning agent becomes intractable [6,7] if even minor variations to the task are introduced. For such settings, the application of deep reinforcement learning requires algorithms that can learn from unshaped, and usually sparse, reward signals. The complicated dynamics of robot manipulation exacerbate the difficulty posed by sparse rewards, especially for on-policy RL algorithms. For example, achieving goals that require successfully executing multiple steps over a long horizon involves high dimensional control that must also generalise to work across variations in the environment for each step. These aspects of robot control result in a situation where a naive RL agent so rarely receives a reward at the start of training that it is not able to learn at all. A common solution in the robotics community is to collect a sufficient quantity of expert demonstrations, then use imitation learning to train the agent. However, in some scenarios, demonstrations are expensive to collect and the achievable performance of a trained agent is restricted by their quantity. One solution is to use the valuable past experiences of the agent to enhance training, and this is particularly useful in sparse reward environments.

To alleviate the problems associated with having sparse rewards, there are two kinds of approaches: imitation learning and hindsight experience replay (HER). First, the standard approach of imitation learning is to use supervised learning algorithms and minimise a surrogate loss with respect to an oracle. The most common form is learning from demonstrations [8,9]. Similar techniques are applied to robot manipulation tasks [10–13]. When the demonstrations are not attainable, self-imitation learning (SIL) [14], which uses past good experiences (episodes in which the goal is achieved), can be used to enhance exploration or speed up the training of the agent. Self-imitation learning works well in discrete control environments, such as Atari Games. Whilst being able to learn policies for continuous control tasks with dense or delayed rewards [14], the present experiments suggest that SIL struggles when rewards are sparse. Recently, hindsight experience replay has been proposed to solve such goal-conditional, sparse reward problems. The main idea of HER [15] is that during replay, the selected transitions are sampled from state–action pairs derived from achieved goals that are substituted for the real goals of the task; this increases the frequency of positive rewards. Hindsight experience replay is used with off-policy RL algorithms, such as DQN [1] and DDPG [16], for experience replay and has several extensions [17,18]. The present experiments show that simply applying HER with SIL does not lead to an agent capable of performing tasks from the Fetch robot environment. In summary, self-imitation learning with on-policy algorithms for tasks that require continuous control, and for which rewards are sparse, remains unsolved.

In this paper, episodic self-imitation learning (ESIL) for goal-oriented problems that provide only sparse rewards is proposed and combined with a state-of-the-art on-policy RL algorithm: proximal policy optimization (PPO). In contrast to standard SIL, which samples past good transitions from the replay buffer for imitation learning, the proposed ESIL adopts entire current episodes (successful or not), and modifies them into “expert” trajectories based on HER. An extra trajectory selection module is also introduced to relieve the effects of sample correlation [19] in updating the network. Figure 1 shows the difference between naive SIL+HER and ESIL. During training by SIL+HER, a batch of transitions is sampled from the replay buffer; these are modified into “hindsight experiences” and used directly in self-imitation learning. In contrast, ESIL utilises entire current collected episodes and converts them into hindsight episodes. The trajectory selection module removes undesired transitions in the hindsight episodes. Using tasks from the Open AI Fetch environment, this paper demonstrates that the proposed ESIL approach is effective in training agents which are required to solve continuous control problems, and shows that it achieves state-of-the-art results on several tasks.

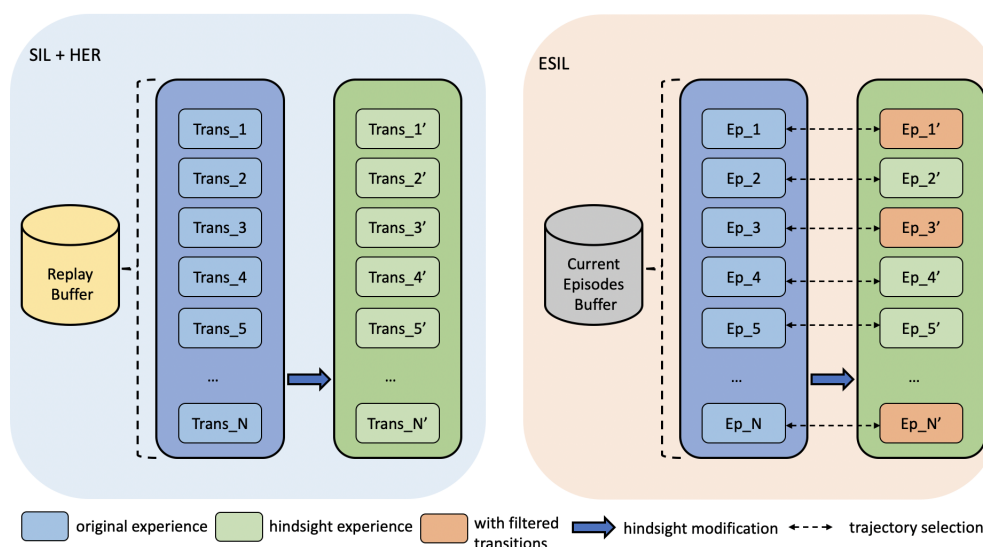


Figure 1. Illustration of difference between self-imitation learning (SIL)+hindsight experience replay (HER) and episodic self-imitation learning (ESIL).

The primary contribution of this paper is a novel episodic self-imitation learning (ESIL) algorithm that can solve continuous control problems in environments providing only sparse rewards; in doing so, it also empirically answers an open question posed by Plappert et al. [20]. The proposed ESIL approach also provides a more efficient way to perform exploration in goal-conditional settings than the standard self-imitation learning algorithm. Finally, this approach achieves, to our knowledge, the best results for four moderately complex robot control tasks in simulation. The paper is organised into the following structure: Sections 2 and 3 provide an introduction to related work and corresponding background. Section 4 describes the methodology of the proposed ESIL approach. Section 5 introduces the settings and results of the experiments. Finally, Section 6 provides concluding remarks and suggestions for future research. The code presenting the implementation of ESIL used in this work is available from here: <https://github.com/TianhongDai/esil-hindsight>.

2. Related Work

Imitation learning (IL) can be divided into two main categories: behavioural cloning and inverse reinforcement learning [21]. Behavioural cloning involves the learning of behaviours from demonstrations [22–24]. Other extensions have an expert in the loop, such as DAgger [25], or use an adversarial paradigm for the behavioural cloning method [26,27]. The inverse reinforcement learning estimates a reward model from expert trajectories [28–30]. Learning from demonstrations is powerful for complex robotic manipulation tasks [10,31–34]. Ho and Ermon [26] propose generative adversarial imitation learning (GAIL), which employs generative adversarial training to match the distribution of state–action pairs of demonstrations. Compared with behavioural cloning, the GAIL framework shows strong improvements in continuous control tasks. In the work of [35], goalGAIL is proposed to speed up the training in goal-conditional environments; goalGAIL was also shown to be able to learn from demonstrations without action information. Prior work has used demonstrations to accelerate learning [10–12]. Demonstrations are often collected by an expert policy or human actions. In contrast to these approaches, episodic self-imitation learning (ESIL) does not need demonstrations.

Self-imitation learning (SIL) [14] is used for exploiting past experiences for parametric policies. It has a similar flavor to [36,37], in that the agent learns from imperfect demonstrations. During training, past good experiences are stored in the replay buffer. When SIL starts, transitions are sampled from the replay buffer according to the advantage values. In the work of Tang [38], generalised SIL was proposed as an extension of SIL. It uses an n -bound Q -learning approach to generalise the original SIL technique, and shows robustness to a wide range of continuous control tasks. Generalised SIL can also be combined with both deterministic and stochastic RL algorithms. Guo et al. [39] points out that using imitation learning with past good experience could lead to a sub-optimal policy. Instead of imitating past good trajectories, a trajectory-conditioned policy [39] is proposed to imitate trajectories in diverse directions, encouraging exploration in environments where exploration is otherwise difficult. Unlike SIL, episodic self-imitation learning (ESIL) applies HER to the current episodes to create “imperfect” demonstrations for imitation learning; this also requires introducing a trajectory-selection module to reject undesired samples from the hindsight experiences. In the work of Lee et al. [19], it was shown that the agent benefits from using whole episodes in updates, rather than uniformly sampling the sparse or delayed reward environments. The present experiments suggest that episodic self-imitation learning achieves better performance in an agent that must learn to perform continuous control in environments delivering sparse rewards.

Recently, the technique known as hindsight learning was developed. Hindsight experience replay (HER) [15] is an algorithm that can overcome the exploration problems in multi-goal environments, delivering sparse rewards. Hindsight policy gradient (HPG) [40] introduces techniques that enable the learning of goal-conditional policies using hindsight experiences. However, the current implementation of HPG has only been evaluated for agents that need to perform discrete actions, and one drawback of hindsight policy gradient estimators is the computational cost because of the goal-oriented sampling. An extension of HER, called dynamic hindsight experience replay (DHER) [41], was proposed to

deal with dynamic goals. Liu et al. [42] uses the GAIL framework [26] to generate trajectories that are similar to hindsight experiences; it then applies imitation learning, using these trajectories. Competitive Experience Replay (CER) complements HER by introducing a competition between two agents for exploration [18]. Zhao and Tresp [43] point out that the hindsight trajectories which contain higher energy are more valuable during training, leading to a more efficient learning system. Fang et al. [44] proposed curriculum-guided HER, which incorporates curriculum learning in the work. During training, the agent focuses on the closest goals in the initial stage, then focuses on the expanding the diversity of goals. This approach accelerates training compared with other baseline methods. Unlike these works, episodic self-imitation learning (ESIL) combines episodic hindsight experiences with imitation learning, which aids learning at the start of training. Furthermore, ESIL can be applied to continuous control, making it more suitable for control problems that demand greater precision.

3. Background

3.1. Reinforcement Learning

Reinforcement Learning (RL) can be formulated under the framework of a Markov Decision Process (MDP); it is used to learn an optimal policy to solve sequential decision-making problems. In each time step t , the state s_t is received by the agent from the environment. An action a_t is sampled by the agent according to its policy $\pi_\theta(s_t|a_t)$, parameterised by θ , which—in deep reinforcement learning—represent the weights of an artificial neural network. Then, the state s_{t+1} and reward r_{t+1} are provided by the environment to the agent. The goal is to have the agent learn a policy that maximises the expected return $\mathbb{E}_\theta[R(s_t, a_t)]$ [45]

$$R(s_t, a_t) = \sum_{l=0}^{T-1} \gamma^l r_{t+l}, \quad (1)$$

where γ is the discount factor. In a robot control setting, the state s_t can be the velocity and position of each joint of the robotic arm. The action a_t can be the velocities of actuators (control signals) and the reward r_t might be calculated based on the distance between the gripper of the robot arm and the target position.

3.2. Proximal Policy Optimization

In this work, proximal policy optimization (PPO) [46] is selected as our base RL algorithm. This is a state-of-the-art, on-policy actor-critic approach to training. The actor-critic architecture is common in deep RL; it is composed of an actor network which is used to output a policy, and a critic network which outputs a value to evaluate the current state, s_t . Proximal policy optimization (PPO) has been widely tested in robot control [47] and video games [48]. In contrast with the “vanilla policy” gradient algorithms, proximal policy optimization (PPO) learns the policy using a surrogate objective function

$$\mathcal{L}_{policy} = \mathbb{E}_t \left[\min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} A_t, \text{clip} \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, 1 - \epsilon, 1 + \epsilon \right) A_t \right) \right], \quad (2)$$

where $\pi_\theta(a_t|s_t)$ is the current policy and $\pi_{\theta_{old}}(a_t|s_t)$ is the old policy; ϵ is a clipping ratio which limits the change between the updated and the previous policy during the training process. A_t is the advantage value which can be estimated as $R(s_t, a_t) - V(s_t)$, with $R(s_t, a_t)$ being the return value, and $V(s_t)$ the state value predicted by the critic network.

3.3. Hindsight Experiences and Goals

The experiments follow the terminology suggested by OpenAI [20], in which the possible goals are drawn from \mathcal{G} , and the goal being pursued does not influence the environment dynamics. In ESIL, two types of goal are recognised. One is the desired goal $g \in \mathcal{G}$, which is the target position or state, and may be different for different episodes. Within a single episode, g is constant. The second type of goal is the achieved goal g^{ac} , which is the achieved state in the environment, and this is considered to be different at each time step in an episode. In an episode, each transition can be represented as $(s_t | \langle g, g_t^{ac} \rangle, a_t, r_t, s_{t+1} | \langle g, g_{t+1}^{ac} \rangle)$, where s_t indicates a state, a_t indicates an action and r_t indicates a reward; \langle, \rangle is simply used to represent grouping of goals.

In sparse reward settings, an agent will only get positive rewards when the desired goal g is achieved. The sparse reward function can be defined as

$$r_t(g_t^{ac}, g) := \begin{cases} 0 & \text{if } \|g_t^{ac} - g\| \leq \epsilon \\ -1 & \text{otherwise,} \end{cases} \quad (3)$$

where ϵ is a threshold value, used to identify if the agent has achieved the goal. However, the desired goal, g , might be difficult to reach during training. Thus, hindsight experiences are created through replacing the original desired goal g with the current achieved goal g_t^{ac} to augment the successful samples, and then reward r_t can be recomputed according to Equation 3. The modification of the desired goal can be denoted as g' and transitions from hindsight experiences can be represented as $(s_t | \langle g', g_t^{ac} \rangle, a_t, r_t, s_{t+1} | \langle g', g_{t+1}^{ac} \rangle)$. Intuitively, introducing g' serves a useful purpose in the early stages of training; taking, for example, a robot reaching task, the agent has no prior concept of how to move its effector to a specific location in space. Thus, even these original failed episodes contain valuable information for ultimately learning a useful control policy for the original, desired goal g .

4. Methodology

The proposed method combines PPO and episodic self-imitation learning to maximally use hindsight experiences for exploration to improve learning. Recent advantages in episodic backward update [19] and hindsight experiences [15] are also leveraged to guide exploration for on-policy RL.

4.1. Episodic Self-Imitation Learning

The present method aims to use episodic hindsight experiences to guide the exploration of the PPO algorithm. To this end, hindsight experiences are created from current episodes. For an episode i , let there be T time steps; after T , a series of transitions $\tau_i = \{(s_t | \langle g, g_t^{ac} \rangle, a_t, r_t, s_{t+1} | \langle g, g_{t+1}^{ac} \rangle)\}_{t=0:T-1}$ is collected. If at time step $t = T - 1$, in s_T , $g_T^{ac} \neq g$, it implies that in this episode, the agent failed to achieve the original goal. Simply, to create hindsight experiences, the achieved goal g_T^{ac} in the last state s_T is selected and considered as the modified desired goal g' , i.e., $g' = g_T^{ac}$. Next, a new reward r'_t is computed under the new goal g' . Then, a new “imagined” episode is achieved, and a new series of transitions $\tau'_i = \{(s_t | \langle g', g_t^{ac} \rangle, a_t, r'_t, s_{t+1} | \langle g', g_{t+1}^{ac} \rangle)\}_{t=0:T-1}$ is collected.

Then, an approach to self-imitation learning based on episodic hindsight experiences is proposed, which applies the policy updates to both hindsight and in-environment episodes. Proximal policy optimization (PPO) is used as the base RL algorithm, which is a state-of-the-art on-policy RL algorithm. With current and corresponding hindsight experiences, a new objective function is introduced and defined as

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{PPO} + \beta \cdot \mathcal{L}_{ESIL}, \quad (4)$$

where α is the weight coefficient of \mathcal{L}_{PPO} . In the experiments, we set $\alpha = 1$ as default to balance the contribution of \mathcal{L}_{PPO} and \mathcal{L}_{ESIL} . \mathcal{L}_{PPO} is the loss of PPO which can be written as

$$\mathcal{L}_{PPO} = \mathcal{L}_{policy} - c \cdot \mathcal{L}_{value} \quad (5)$$

where \mathcal{L}_{policy} is the policy loss which is parameterised by θ , \mathcal{L}_{value} is the value loss which is parameterised by η , and c is the weight coefficient of the \mathcal{L}_{value} , which is set to 1 to match the default PPO setting [46]. The policy loss, \mathcal{L}_{policy} , can be represented as

$$\mathcal{L}_{policy} = \mathbb{E}_{s_t, a_t, g \in \mathcal{T}} \left[\min \left(\frac{\pi_{\theta}(a_t | s_t, g)}{\pi_{\theta_{old}}(a_t | s_t, g)} A_t, \text{clip} \left(\frac{\pi_{\theta}(a_t | s_t, g)}{\pi_{\theta_{old}}(a_t | s_t, g)}, 1 - \epsilon, 1 + \epsilon \right) A_t \right) \right], \quad (6)$$

here, A_t is the advantage value, and can be computed as $R_t - V_{\eta}(s_t, g)$. $V_{\eta}(s_t, g)$ is the state value at time step t which is predicted by the critic network. R_t is the return at time step t . ϵ is the clip ratio. \mathcal{T} indicates original trajectories. The value loss is an squared error loss $\mathcal{L}_{value} = (V_{\eta}(s_t, g) - R_t)^2$.

For the \mathcal{L}_{ESIL} term, β is an adaptive weight coefficient of \mathcal{L}_{ESIL} ; it can be defined as the ratio of samples which are selected for self-imitation learning

$$\beta = \frac{N_{ESIL}}{N_{Total}}, \quad (7)$$

where N_{ESIL} is the number of samples used for self-imitation learning and N_{Total} is the total number of collected samples. The episodic self-imitation learning loss \mathcal{L}_{ESIL} can be written as

$$\mathcal{L}_{ESIL} = \mathbb{E}_{s_t, a_t, g' \in \mathcal{T}'} [\log \pi_{\theta}(a_t | s_t, g') \cdot \mathcal{F}_t], \quad (8)$$

where \mathcal{T}' indicates hindsight trajectories and \mathcal{F}_t is the trajectory selection module which is based on returns of the current episodes, R , and the returns of corresponding hindsight experiences, R' .

4.2. Episodic Update with Hindsight

Two important issues of ESIL are: (1) hindsight experiences are sub-optimal, and (2) the detrimental effect of updating networks with correlated trajectories. Although episodic self-imitation learning makes exploration more effective, hindsight experiences are not from experts and not “perfect” demonstrations. With the training process continuing, if the agent is always learning these imperfect demonstrations, the policy will be stuck at the sub-optimal, or experience overfitting.

To prevent the agent learning from imperfect hindsight experiences, hindsight experiences are actively selected based on returns. With the same action, different goals may lead to different results. The proposed method only selects hindsight experiences that can achieve higher returns. The illustration of the trajectory selection module is in Figure 2. For an episodic experience and its hindsight experience, the returns of the episodic experience and its hindsight experience can be calculated, respectively. In a trajectory, at time step t , the return R_t can be calculated by $R_t = R(s_t, g) = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \dots + \gamma^{T-t-1} \cdot r_{T-1}$. Then, for a trajectory τ_i , we have $\{R_0^i, R_1^i, R_2^i, \dots, R_{T-1}^i\}$. For the hindsight experiences, similarly, the return R'_t for each time step, t , with respect to the hindsight goals g' , can be calculated. Based on the modified trajectory τ'_i with the same length of τ_i , we therefore have the returns $\{R_0^{i'}, R_1^{i'}, R_2^{i'}, \dots, R_{T-1}^{i'}\}$. During training, the hindsight experiences with higher returns are used for self-imitation learning. The rest of the hindsight experiences will be supposed to be worthless samples and ignored. Then, Equation (8) can be rewritten as

$$\mathcal{L}_{ESIL} = \mathbb{E}_{s_t, a_t, g' \in \mathcal{T}', g \in \mathcal{T}} \left[\log \pi_{\theta}(a_t | s_t, g') \cdot \mathcal{F}(s_t, g, g') \right], \quad (9)$$

where $\mathcal{F}(s_t, g, g')$ is the trajectory selection module. The selection function can be expressed as

$$\mathcal{F}(s_t, g, g') = \mathbb{1}[R(s_t, g') > R(s_t, g)], \quad (10)$$

here, $\mathbb{1}(\cdot)$ is the unit step function. Consider the OpenAI FetchReach environment as an example. For a failed trajectory, the rewards r_t are $\{-1, -1, \dots, -1\}$. The desired goal is modified to construct

a new hindsight trajectory and the new rewards r'_t become $\{-1, -1, \dots, 0\}$. Then, R and R' can be calculated separately.

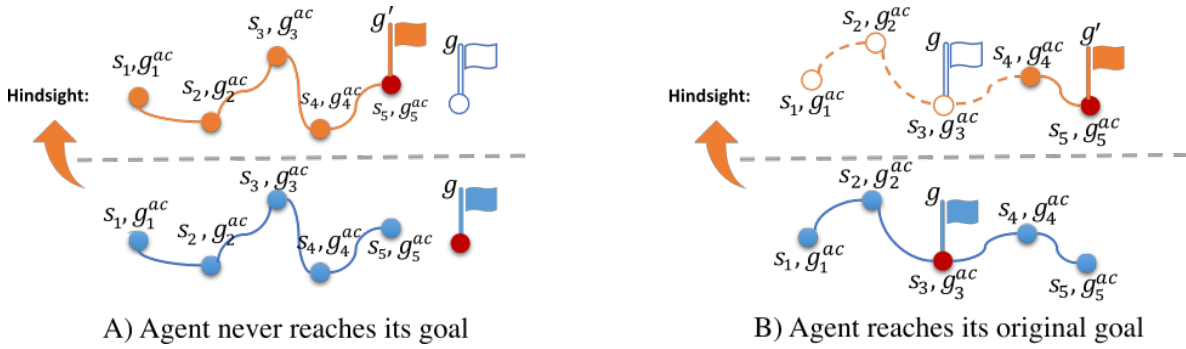


Figure 2. A simplified illustration of trajectory selection. Blue trajectories indicate original experiences. Orange trajectories indicate hindsight experiences. Solid trajectories in the hindsight experiences are selected by the trajectory selection module of ESIL with new “imagined” goals.

From a goal perspective, episodic self-imitation learning (ESIL) tries to explore (desired) goals to get positive returns. It can be viewed as a form of multi-task learning, because ESIL has two objective functions to be optimised jointly. It is also related to self-imitation learning (SIL) [14]. However, the difference is that SIL uses $(R - V_\theta(s))_+$ on past experiences to learn to choose the action chosen in the past in a given state, rather than goals. The full description of ESIL can be found in Algorithm 1.

Algorithm 1 Proximal policy optimization (PPO) with Episodic Self-Imitation Learning (ESIL)

Require: an actor network $\pi(s, g|\theta)$, a critic network $V(s, g|\eta)$, the maximum steps T of an episode, a reward function r

```

1: for iteration = 1, 2, ... do
2:    $\mathcal{T} = \emptyset, \mathcal{T}' = \emptyset$ 
3:   for episode = 1, 2, ...,  $N$  do
4:      $\tau = \emptyset$ 
5:     for  $t = 0, 1, \dots, T - 1$  do
6:       Sample an action  $a_t$  using the actor network  $\pi(s_t, g|\theta)$ 
7:       Execute the action  $a_t$  and observe a new state  $s_{t+1}$ 
8:       Store the transition  $(s_t | \langle g, g_t^{ac} \rangle, a_t, r_t, s_{t+1} | \langle g, g_{t+1}^{ac} \rangle)$  in  $\tau$ 
9:     end for
10:    for each transition  $(s_t, a_t, r_t, g, g_t^{ac})$  in  $\tau$  do
11:      Clone the transition and replace  $g$  with  $g'$ , where  $g' = g_t^{ac}$ 
12:       $r'_t := r(s_t, a_t, g')$ 
13:      Store the transition  $(s_t, a_t, r'_t, g', g_t^{ac})$  in  $\tau'$ 
14:    end for
15:    Store the trajectory  $\tau$  and the hindsight trajectory  $\tau'$  in  $\mathcal{T}$  and  $\mathcal{T}'$ , respectively
16:  end for
17:  Calculate the Return  $R$  and  $R'$  for all transitions in  $\mathcal{T}$  and  $\mathcal{T}'$ , respectively
18:  Calculate the PPO loss:  $\mathcal{L}_{PPO} = \mathcal{L}_{policy}(\theta) - c \cdot \mathcal{L}_{value}(\eta)$  using  $\mathcal{T}$  (5)
19:  Calculate the ESIL loss:  $\mathcal{L}_{ESIL}(\theta)$  using  $\mathcal{T}'$ ,  $R$  and  $R'$  (8)
20:  Update the parameters  $\theta$  and  $\eta$  using loss  $\mathcal{L} = \alpha \cdot \mathcal{L}_{PPO} + \beta \cdot \mathcal{L}_{ESIL}$  (4)
21: end for

```

5. Experiments and Results

The proposed method is evaluated on several multi-goal environments, including the Empty Room environment and the OpenAI Fetch environments (see Figure 3). The Empty Room environment is a toy example, and has discrete action spaces. In the Fetch environments, there are four robot tasks

with continuous action spaces. To obtain a comprehensive comparison between the proposed method and other baseline approaches, suitable baseline approaches are selected for different environments. Ablation studies of the trajectory selection module are also performed. Our code and models will be publicly available.

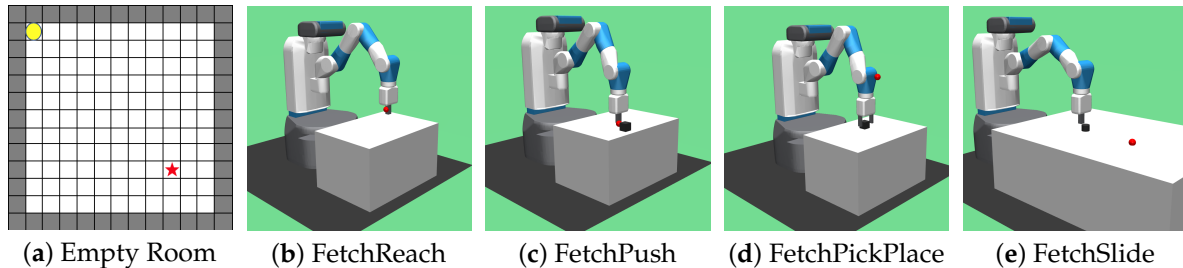


Figure 3. Evaluation environments. (a) is the Empty Room environment, in which a yellow circle indicates the position of the agent and a red star represents a target position. (b–e) are the Fetch robotic environments. The red spot represents a target position.

5.1. Setup

Empty Room (grid-world) environment: The Empty Room environment is a simple grid-world environment. The agent is placed in an 11×11 grid, representing the room. The goal of the agent is to reach a target position in the room. The start position of the agent is at the left upper corner of the room, and the target position is randomly selected within the room. When the agent chooses an action that would lead it to fall outside the grid area, the agent stays at the current position. The length of each episode is 32. The desired goal, g , is a two-dimensional grid coordinate which represents the target position. The achieved goal, g_t^{ac} , is also a two-dimensional coordinate which represents the current position of the agent at time step t , and finally, the observation is a two-dimensional coordinate which represents the current position of the agent. The agent has five actions: left, right, up, down and stay; the agent executes a random action with probability 0.2. The agent can get +1 as a reward only when $g_t^{ac} = g$, otherwise, it gets a reward of 0.

The agent is trained with 1 CPU core. In each epoch, 100 episodes are collected for the training. After each epoch, the agent is evaluated for 10 episodes. During training, the actions are sampled from the categorical distribution. During evaluation, the action with the highest probability will be chosen.

Fetch robotic (continuous) environments [20]: The Fetch robotic environments are physically plausible simulations based on the real Fetch robot. The purpose of these environments is to provide a platform to tackle problems which are close to practical challenging robot manipulation tasks. Fetch is a 7-DoF robot arm with a two finger gripper. The Fetch environments include four tasks: *FetchReach*, *FetchPush*, *FetchPickAndPlace* and *FetchSlide*. For all Fetch tasks, the length of each episode is 50. The desired goal, g , is a three-dimensional coordinate which represents the target position. If a task has an object, the achieved goal g_t^{ac} is a three-dimensional coordinate represents the position of the object. Otherwise, g_t^{ac} is a three-dimensional coordinate represents the position of the gripper. Observations include the following information: position, velocity and state of the gripper. If a task has an object, the position, velocity and rotation information of the object is included. Therefore, the observation of *FetchReach* is a 10-dimensional vector. The observation of other tasks is a 25-dimensional vector. The action is a four-dimensional vector. The first three dimensions represent the relative position that the gripper needs to move in the next step. The last dimension indicates the distance between the fingers of the gripper. The reward function can be written as $r_t = -\mathbb{1}(\|g_t^{ac} - g\| > \epsilon)$, where $\epsilon = 0.05$.

In the Fetch environments, for *FetchReach*, *FetchPush* and *FetchPickAndPlace* tasks, the agent is trained using 16 CPU cores. In each epoch, 50 episodes are collected for training. The *FetchSlide* task is more complex, so 32 CPU cores are used. In each epoch, 100 episodes are collected for training. The Message Passing Interface (MPI) framework is used to perform synchronization when

updating the network. After each epoch, the agent is evaluated for 10 episodes by each MPI worker. Finally, the success rate of each MPI worker is averaged. During training, actions are sampled from multivariate normal distributions. In the evaluation phase, the mean vector of the distribution is used as an action.

The proposed method, termed PPO+ESIL, is compared with different baselines on different environments. All experiments are plotted based on five runs with different seeds. The solid line is the median value. The upper bound is the 75th percentile and the lower bound is the 25th percentile.

5.2. Network Structure and Hyperparameters

Network structure: Both the actor network and the critic network have three hidden layers with 256 neurons. ReLu is selected as the activation function for the hidden layers. In the grid-world environment, the actor network builds a categorical distribution. In the Fetch environment, the actor network builds normal distributions by producing mean vectors and the standard deviations of the independent variables.

Hyperparameters: For all experiments, the learning rate is 0.0003 for both the actor and critic networks. The discount factor γ is 0.98. Adam is chosen as an optimiser with $\epsilon = 0.00001$. For each epoch, the actor network and critic network are updated 10 times. The clip ratio of the PPO algorithm is 0.2. For the grid-world environment, it trains networks for 100 epochs with batch size equals 160. Each epoch consists of 100 episodes. For the Fetch environments, in FetchReach task, it trains networks for 100 epochs and other tasks for 1000 epochs with batch size equals to 125. For FetchReach, FetchPush and FetchPickAndPlace tasks, each epoch consists of 50 episodes. For FetchSlide task, each epoch consists of 100 episodes. In designing the experiments, the number of episodes within an epoch is a balance between being able to train, the length of time required to run experiments and the maximum number of time steps that would be required to achieve a goal. All environments have a fixed maximum number of time-steps, but this maximum differs depending on the problem or environment. This means that the number of state-action pairs can differ between two environments that have the same number of episodes and the same number of epochs. We arrange the episodes to try to compensate for the number of state-action pairs collected during training to make experiments easier to compare. The models are trained on a machine with an Intel i7-5960X CPU and 64GB RAM.

5.3. Grid-World Environments

To understand the basic properties of the proposed method, the toy Empty Room environment is used to evaluate ESIL. The following baselines are considered:

- PPO: vanilla PPO [46] for discrete action spaces;
- PPO+SIL/PPO+SIL+HER: Self-imitation learning (SIL) is used with PPO to solve hard exploration environments by imitating past good experiences [14]. In order to solve sparse rewards tasks, hindsight experience replay (HER) is applied to sampled transitions;
- DQN+HER: Hindsight experience replay (HER), designed for sparse reward problems, is combined with a deep Q-learning network (DQN) [15]; this is an off policy algorithm;
- Hindsight Policy Gradients (HPG): the vanilla implementation of HPG that is only suitable for discrete action spaces [40].

More specifically, PPO+ESIL is compared with above baseline methods in Figure 4a. This shows that PPO+ESIL converges faster than the other four baselines, and PPO+SIL converges faster than vanilla PPO, because PPO+SIL reuses past good experiences to help exploration and training. Hindsight Policy Gradient (HPG) is slower than the others because goal sampling is not efficient and also unstable.

Further, the performance of the trajectory selection module is evaluated in Figure 4b. This shows that the selection strategy helps improve the performance. Hindsight experiences are not always perfect; the trajectory selection module filters some undesirable, modified experiences. Through

adopting this selection strategy, the chance of agents learning from poor trajectories is reduced. The adaptive weight coefficient β is also investigated in these experiments. In Figure 4c, it can be seen that at the initial stages of training, β is high. This is because at this stage, the agent very seldom achieves the original goals. The hindsight experiences can yield higher returns than the original experiences. Therefore, a large proportion of hindsight experiences are selected to conduct self-imitation learning, helping the agent learn a policy for moving through the room. In the later stages of training, the agent can achieve success frequently, and the hindsight experiences might be redundant (e.g., $R(s_t, g) \geq R(s_t, g')$). In this case, undesired hindsight experiences are removed by using the trajectory selection module and L_{PPO} leads the training. However, when the trajectory selection module is not employed, all hindsight experiences are used through the entire training process which includes the redundant hindsight experiences. This leads to overfitting and makes training unstable. Thus, the L_{ESIL} can provide the agent with a better initial policy, and the adaptive weight coefficient β can balance the contributions of L_{PPO} and L_{ESIL} properly during training.

Finally, the combination of PPO+ESIL is also compared with DQN+HER, which is an off-policy RL algorithm, in Figure 4d. This shows that DQN+HER works a little better than ESIL at the start of training. However, the proposed method achieves similar results to DQN+HER later in training.

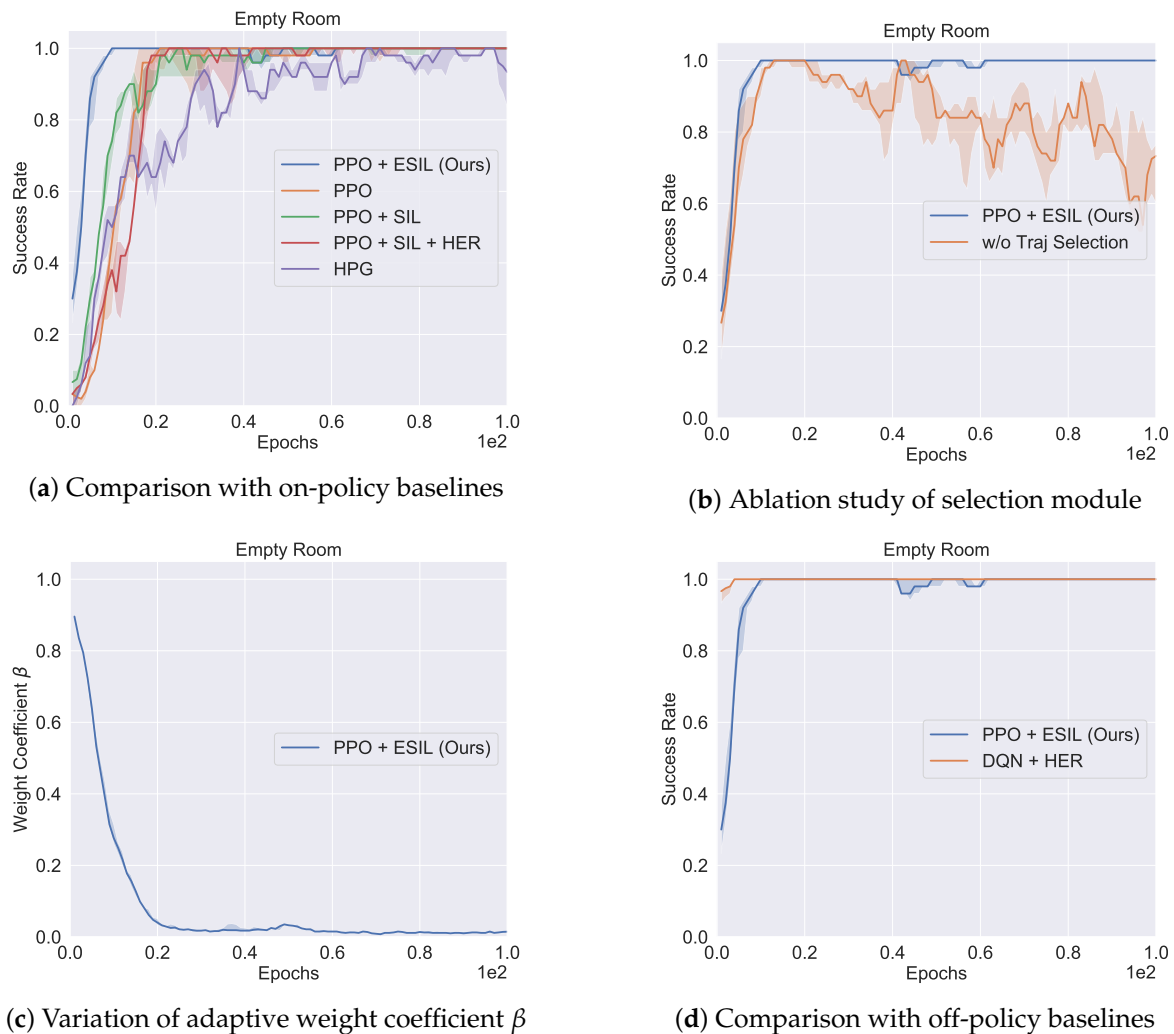


Figure 4. Results of the grid-world environment. (a) Comparing the performance of PPO+ESIL between the on-policy approaches. (b) An ablation study on the trajectory selection module. (c) The variation of adaptive weight coefficient β through training. (d) Comparison of the performance of PPO+ESIL to an off-policy approach: DQN+HER.

5.4. Continuous Environments

Continuous control problems are generally more challenging for reinforcement learning. In the experiments of this section, the aim is to investigate how useful the proposed method is for several hard exploration OpenAI Gym Fetch tasks. These environments are commonly used to assess the performance of RL methods for continuous control. The following baselines are considered:

- PPO: the vanilla PPO [46] for continuous action spaces;
- PPO+SIL/PPO+SIL+HER: Self-imitation learning is used with PPO to solve hard exploration environments by imitating past good experiences [14]. For sparse rewards tasks, hindsight experience replay (HER) is applied to sampled transitions;
- DDPG+HER: this is the state-of-the-art off-policy RL algorithm for the Fetch tasks. Deep deterministic policy gradient (DDPG) is trained with HER to deal with the sparse reward problem [15].

5.4.1. Comparison to On-Policy Baselines

Figure 5, PPO+ESIL achieves reasonable results on all Fetch environments. In contrast, PPO, PPO+SIL and PPO+SIL+HER do not work on all tasks, with the exception of FetchReach. In comparison with the other selected tasks from the Fetch environments, FetchReach is relatively simple, because there is no object to be manipulated. For other tasks, it is quite difficult for the agent to achieve sufficient positive rewards during exploration, because of their rare occurrence. Although PPO+SIL utilises past good experiences to help exploration, it is still faced with the difficulty that past experiences do not easily achieve positive rewards. From the experiments (see Figure 5), PPO+SIL (no hindsight) converges much more slowly than using PPO only. Attempting to use only the original trajectories for self-imitation learning leads to unsatisfactory performance. For PPO+SIL+HER (with no episodic update), the sampled transitions are modified into hindsight experiences, achieving better performance in the FetchReach and FetchSlide tasks. However, this transition-based method still cannot solve the other two manipulation tasks. In contrast, the proposed PPO+ESIL, through utilizing episodic hindsight experiences from failed trajectories, can achieve positive rewards quickly at the start of training.

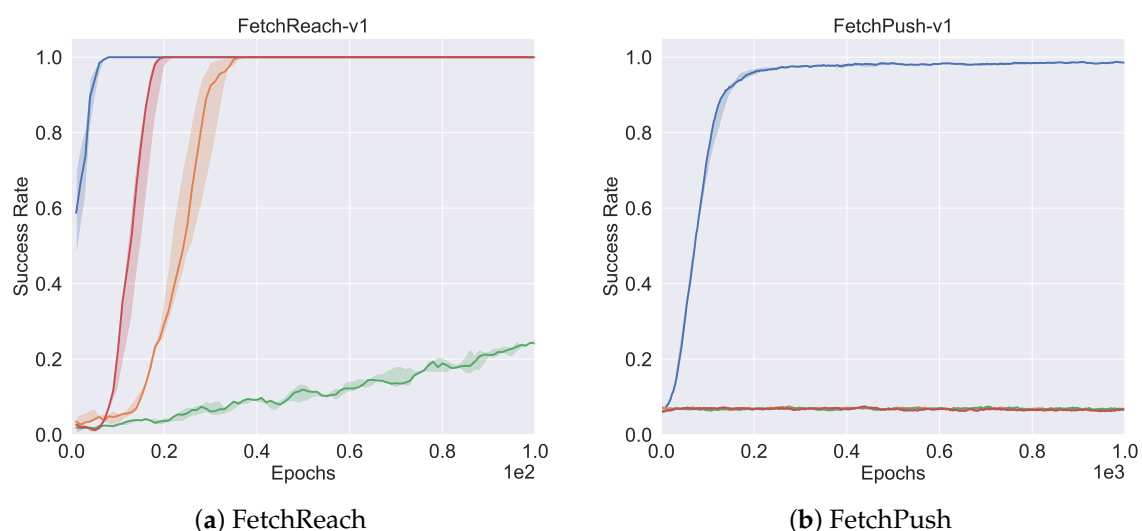


Figure 5. Cont.

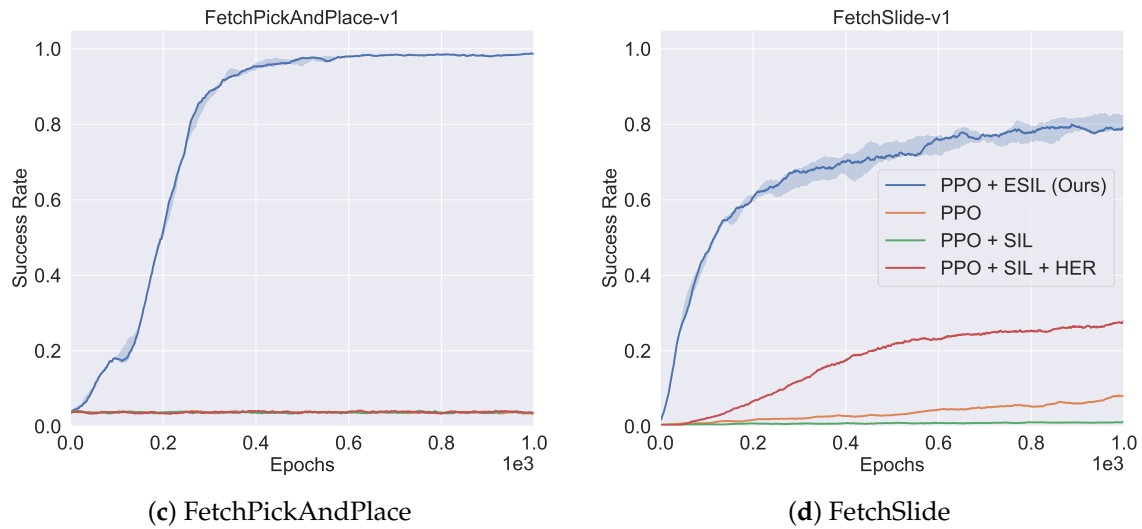


Figure 5. Results of comparison between ESIL and on-policy baselines on all Fetch environments.

5.4.2. Ablation Study of Trajectory Selection Module

In order to investigate the effect of trajectory selection, ablation studies are performed to validate the selection strategy of our approach. Figure 6, when the trajectory selection module is not used, the performance of the agent increases at first, and then starts to decrease. This suggests that the agent starts to converge to a sub-optimal location. However, Figure 6d, for the FetchSlide task, the agent converges faster without the trajectory selection module, and has better performance. This is likely to be because FetchSlide is the most difficult of the Fetch environments. During training, the agent is very unlikely to achieve positive rewards. Figure 7 also indicates that the value of β in FetchSlide is higher than values in other environments, which means the majority of hindsight experiences have higher returns than original experiences. Thus, using *more* hindsight experiences (without filtering) accelerates training at this stage. Nonetheless, the trajectory selection module prevents the agent overfitting the hindsight experience in the other three tasks. Figure 7, shows the adaptive weight coefficient β on all Fetch environments. When the trajectory selection module is used, the value of β decreases with the increase in training epochs. This implies that the agent can achieve a greater proportion of the original goals in the latter stages of training, and fewer hindsight experiences are required for self-imitation learning.

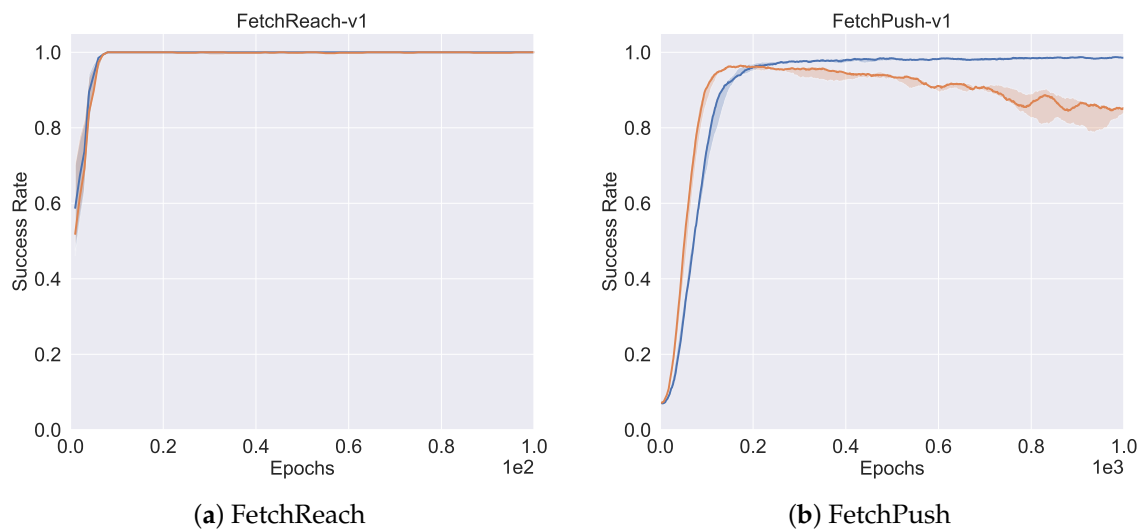


Figure 6. Cont.

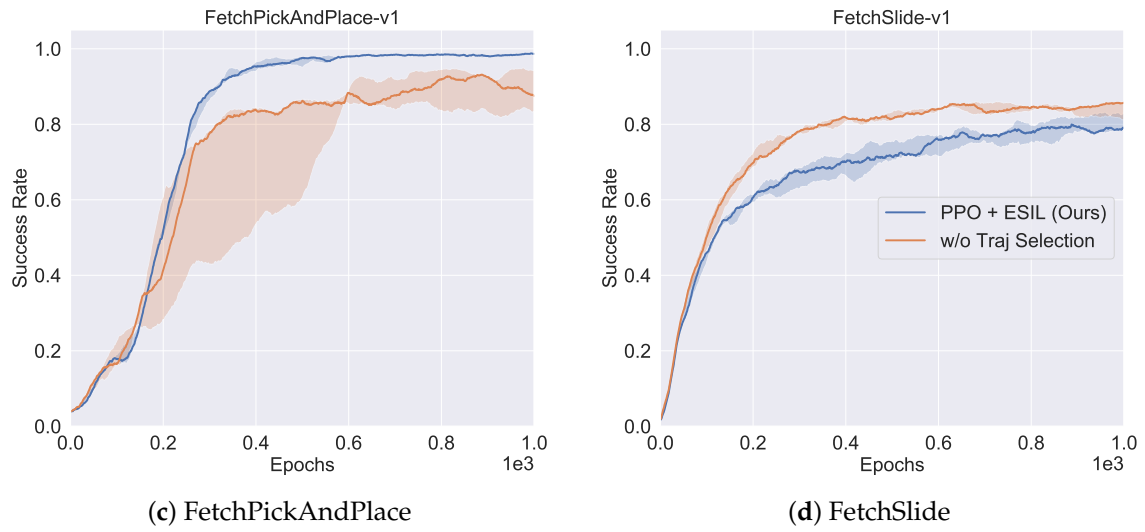


Figure 6. Results of ablation studies with or without using trajectory selection module on all Fetch environments.

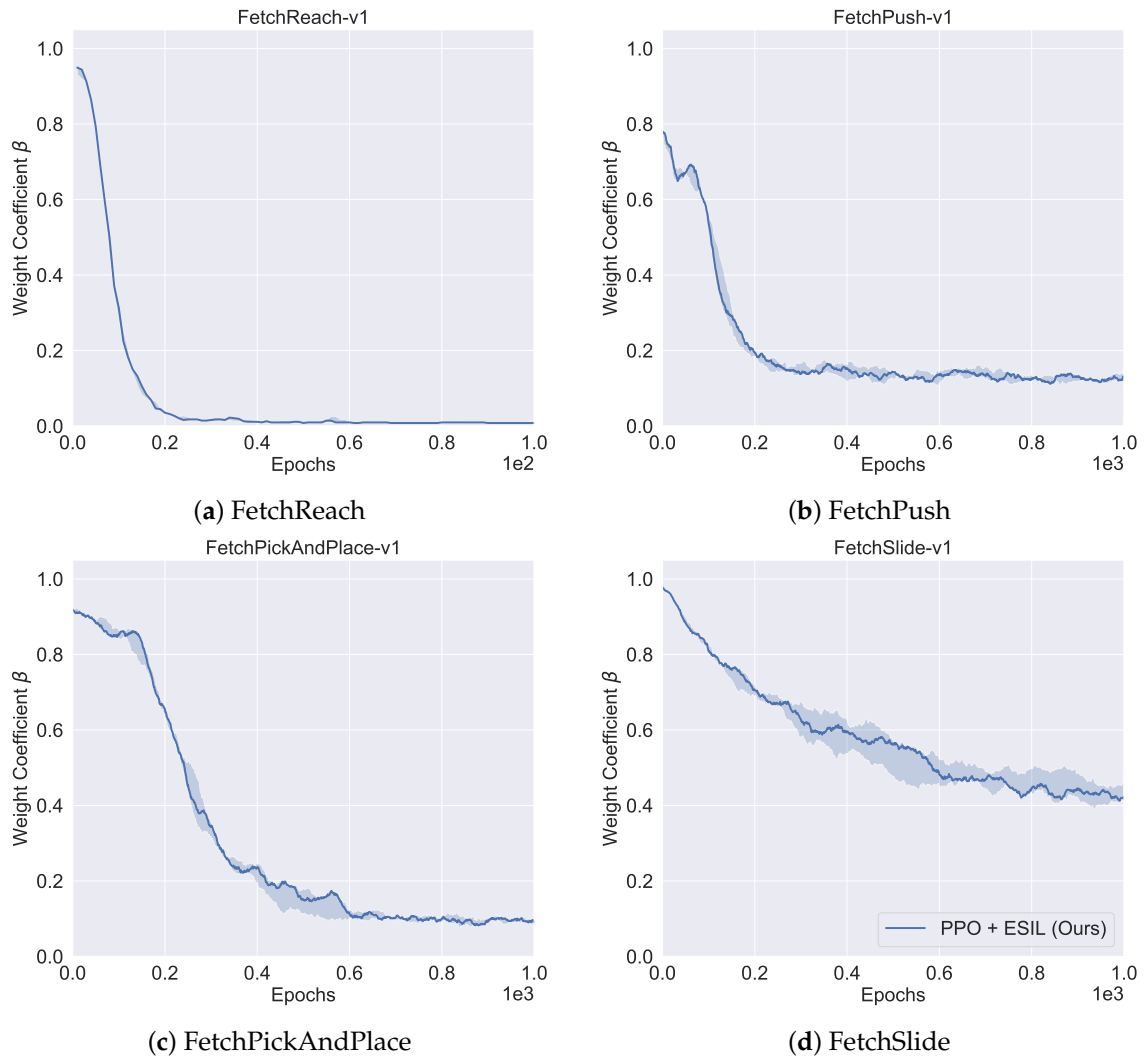


Figure 7. Variation in adaptive weight coefficient β through the training on all Fetch environments.

5.4.3. Comparison to Off-Policy Baselines

Finally, the proposed method is also compared with a state-of-the-art off-policy algorithm: DDPG+HER. From Figure 8, it may be seen that DDPG+HER converges faster than PPO+ESIL in all tasks. However, PPO+ESIL obtains a similar performance to DDPG+HER. This is because DDPG+HER is an off-policy algorithm and uses a large number of hindsight experiences. A replay buffer is also employed to store samples collected in the past. This approach has better sample efficiency than on-policy algorithms such as PPO. Even so, Figure 8c shows that PPO+ESIL still outperforms DDPG+HER in the FetchPickAndPlace task and the success rate is close to 1. This suggests that PPO+ESIL approximates the characteristics of on-policy algorithms, which have low sample efficiency, but are able to obtain a comparable performance to off-policy algorithms in continuous control tasks [46].

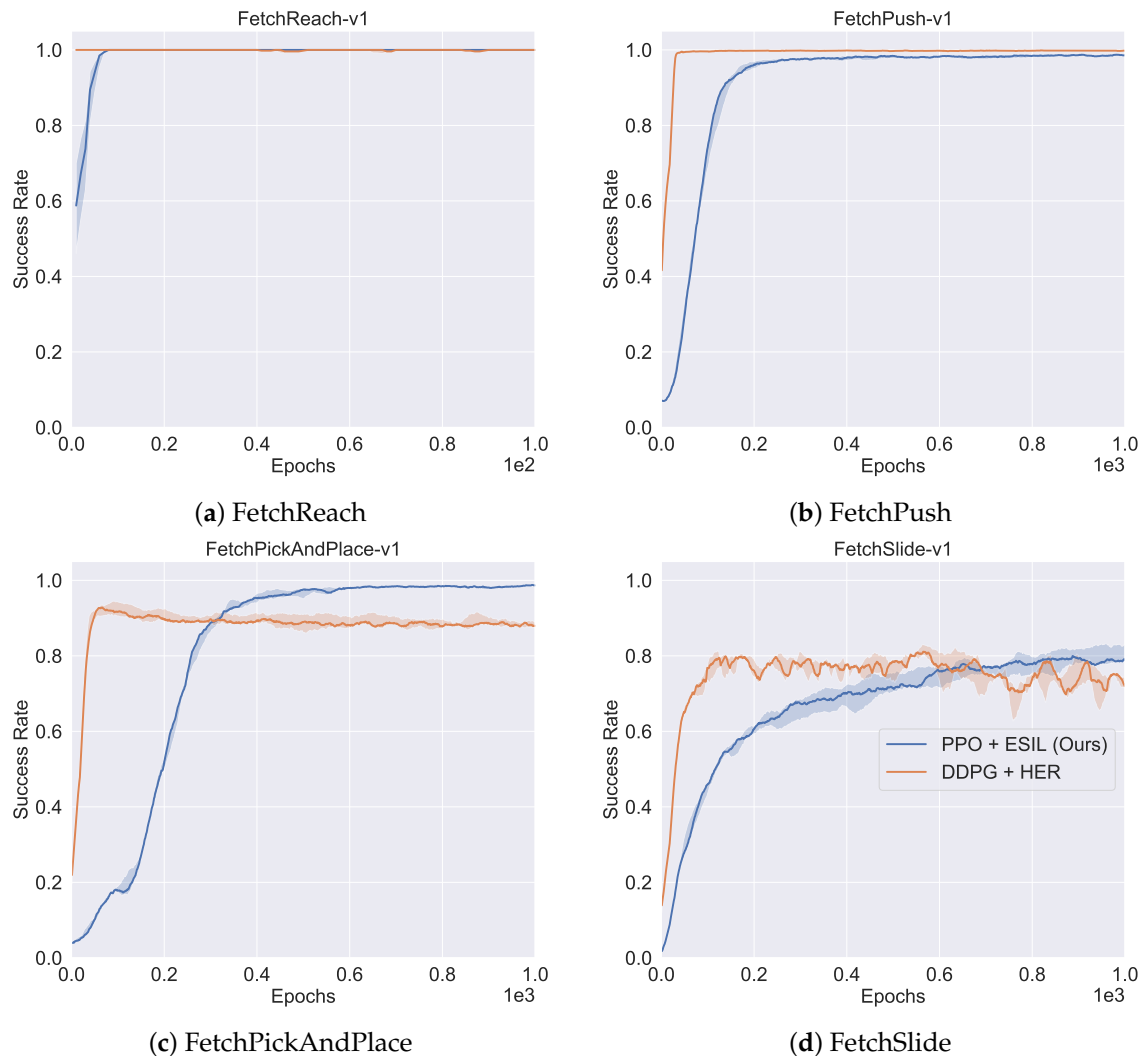


Figure 8. Results of comparison between PPO+ESIL and DDPG+HER on all Fetch environments.

5.5. Overall Performance

Table 1 shows the average success rate of the last 10 epochs during training of baseline methods and PPO+ESIL. The proposed ESIL achieves the best performance in four out of five tasks. However PPO and PPO+SIL only obtain reasonable results for the Empty Room and FetchReach tasks. With the assistance of HER, PPO+SIL+HER obtains a better performance in the FetchSlide task. For the off-policy methods of DDPG+HER, all five tasks are achieved, but a better performance is obtained than PPO+ESIL only in the FetchPush task.

Table 1. Average success rate \pm standard error in the last 10 epochs over five random seeds on all environments (**bold** indicates the best result among all methods).

	Empty Room	Reach	Push	Pick	Slide
PPO	1.000 \pm 0.000	1.000 \pm 0.000	0.070 \pm 0.001	0.033 \pm 0.001	0.077 \pm 0.001
PPO+SIL	0.998 \pm 0.002	0.225 \pm 0.016	0.071 \pm 0.001	0.036 \pm 0.002	0.011 \pm 0.001
PPO+SIL+HER	0.996 \pm 0.013	1.000 \pm 0.000	0.066 \pm 0.011	0.035 \pm 0.004	0.276 \pm 0.011
DQN+HER	1.000 \pm 0.000	-	-	-	-
DDPG+HER	-	1.000 \pm 0.000	0.996 \pm 0.001	0.888 \pm 0.008	0.733 \pm 0.013
HPG	0.964 \pm 0.012	-	-	-	-
PPO+ESIL (Ours)	1.000 \pm 0.000	1.000 \pm 0.000	0.984 \pm 0.003	0.986 \pm 0.002	0.812 \pm 0.015

6. Conclusions

This paper proposed a novel method for self-imitation learning (SIL), in which an on-policy RL algorithm uses episodic modified past trajectories, i.e., hindsight experiences, to update policies. Compared with standard self-imitation learning, episodic self-imitation learning (ESIL) has a better performance in continuous control tasks where rewards are sparse. As far as we know, it is also the first time that hindsight experiences have been combined with state-of-the-art on-policy RL algorithms, such as PPO, to solve relatively hard exploration environments in continuous action spaces.

The experiments that we have conducted suggest that simply using self-imitation learning with the PPO algorithm, even with hindsight experience, leads to disappointing performance in continuous control Fetch tasks. In contrast, the episodic approach we take with ESIL is able to learn in these sparse reward settings. The auxiliary trajectory selection module and the adaptive weight β help the training process to remove undesired experiences and balance the contributions to learning between the PPO term and the ESIL term automatically, and also increase the stability of training.

Our experiments suggest that the selection module is useful to prevent overfitting to sub-optimal hindsight experiences, but also that it does not always lead to learning a better policy faster. Despite this, selection filtering appears to support learning a useful policy in challenging environments. The experiments we have conducted to date have utilised relatively small networks, and it would be appropriate to extend the experiments to consider more complex observation spaces, and to actor/critic networks, which are consequently more elaborate.

Future work includes extending the proposed method to support hierarchical reinforcement learning (HRL) algorithms for more complex manipulation control tasks, such as in-hand manipulation. Episodic self-imitation learning (ESIL) can also be applied to simultaneously learn sub-goal policies.

Author Contributions: Conceptualization, T.D.; methodology, T.D.; software, T.D.; writing—original draft preparation, T.D.; writing—review and editing, H.L. and A.A.B.; supervision, A.A.B.; All authors have read and agreed to the published version of the manuscript.

Funding: This work was partly supported by the Engineering and Physical Sciences Research Council [grant number: EP/J021199/1].

Conflicts of Interest: The authors declare no conflict of interest.

References

1. 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. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529.
2. 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. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484.
3. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436.
4. Liu, W.; Wang, Z.; Liu, X.; Zeng, N.; Liu, Y.; Alsaadi, F.E. A survey of deep neural network architectures and their applications. *Neurocomputing* **2017**, *234*, 11–26.

5. Ng, A.Y.; Harada, D.; Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*; ACM: New York, NY, USA, 1999; Volume 99, pp. 278–287.
6. Arulkumaran, K.; Deisenroth, M.P.; Brundage, M.; Bharath, A.A. Deep Reinforcement Learning: A Brief Survey. *IEEE Signal Process. Mag.* **2017**, *34*, 26–38.
7. Florensa, C.; Held, D.; Wulfmeier, M.; Zhang, M.; Abbeel, P. Reverse curriculum generation for reinforcement learning. *arXiv* **2017**, arXiv:1707.05300.
8. Hester, T.; Vecerik, M.; Pietquin, O.; Lanctot, M.; Schaul, T.; Piot, B.; Horgan, D.; Quan, J.; Sendonaris, A.; Osband, I.; et al. Deep Q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, New Orleans, LA, USA, 2–7 February 2018.
9. Gao, Y.; Lin, J.; Yu, F.; Levine, S.; Darrell, T. Reinforcement learning from imperfect demonstrations. *arXiv* **2018**, arXiv:1802.05313.
10. Rajeswaran, A.; Kumar, V.; Gupta, A.; Vezzani, G.; Schulman, J.; Todorov, E.; Levine, S. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv* **2017**, arXiv:1709.10087.
11. Večerík, M.; Hester, T.; Scholz, J.; Wang, F.; Pietquin, O.; Piot, B.; Heess, N.; Rothörl, T.; Lampe, T.; Riedmiller, M. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv* **2017**, arXiv:1707.08817.
12. Nair, A.; McGrew, B.; Andrychowicz, M.; Zaremba, W.; Abbeel, P. Overcoming exploration in reinforcement learning with demonstrations. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Brisbane, Australia, 21–25 May 2018; pp. 6292–6299.
13. James, S.; Bloesch, M.; Davison, A.J. Task-Embedded Control Networks for Few-Shot Imitation Learning. In *Proceedings of the 2nd Annual Conference on Robot Learning*, Zürich, Switzerland, 29–31 October 2018.
14. Oh, J.; Guo, Y.; Singh, S.; Lee, H. Self-Imitation Learning. In *Proceedings of the International Conference on Machine Learning*, Stockholm, Sweden, 10–15 July 2018.
15. Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Abbeel, O.P.; Zaremba, W. Hindsight experience replay. In *Proceedings of the Advances in Neural Information Processing Systems*, Long Beach, CA, USA, 4–9 December 2017; pp. 5048–5058.
16. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
17. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized experience replay. *arXiv* **2015**, arXiv:1511.05952.
18. Liu, H.; Trott, A.; Socher, R.; Xiong, C. Competitive experience replay. In *Proceedings of the International Conference on Learning Representations*, New Orleans, LA, USA, 6–9 May 2019.
19. Lee, S.Y.; Sungik, C.; Chung, S.Y. Sample-efficient deep reinforcement learning via episodic backward update. In *Proceedings of the Advances in Neural Information Processing Systems*, Vancouver, BC, Canada, 8–14 December 2019; pp. 2112–2121.
20. Plappert, M.; Andrychowicz, M.; Ray, A.; McGrew, B.; Baker, B.; Powell, G.; Schneider, J.; Tobin, J.; Chociej, M.; Welinder, P.; et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv* **2018**, arXiv:1802.09464.
21. Hussein, A.; Gaber, M.M.; Elyan, E.; Jayne, C. Imitation learning: A survey of learning methods. *ACM Comput. Surv.* **2017**, *50*, 21.
22. Bojarski, M.; Del Testa, D.; Dworakowski, D.; Firner, B.; Flepp, B.; Goyal, P.; Jackel, L.D.; Monfort, M.; Muller, U.; Zhang, J.; et al. End to end learning for self-driving cars. *arXiv* **2016**, arXiv:1604.07316.
23. Xu, H.; Gao, Y.; Yu, F.; Darrell, T. End-to-end learning of driving models from large-scale video datasets. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, 21–26 July 2017; pp. 2174–2182.
24. Torabi, F.; Warnell, G.; Stone, P. Behavioral cloning from observation. *arXiv* **2018**, arXiv:1805.01954.
25. Ross, S.; Gordon, G.; Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, Fort Lauderdale, FL, USA, 11–13 April 2011; pp. 627–635.

26. Ho, J.; Ermon, S. Generative adversarial imitation learning. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 4565–4573.
27. Wang, Z.; Merel, J.S.; Reed, S.E.; de Freitas, N.; Wayne, G.; Heess, N. Robust imitation of diverse behaviors. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 5320–5329.
28. Ng, A.Y.; Russell, S.J. Algorithms for Inverse Reinforcement Learning. In Proceedings of the International Conference on Machine Learning, Stanford, CA, USA, 29 June–2 July 2000; pp. 663–670.
29. Abbeel, P.; Ng, A.Y. Apprenticeship learning via inverse reinforcement learning. In Proceedings of the International Conference on Machine Learning, Louisville, KY, USA, 16–18 December 2004; p. 1.
30. Ziebart, B.D.; Maas, A.L.; Bagnell, J.A.; Dey, A.K. Maximum entropy inverse reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Chicago, IL, USA, 13–17 July 2008; Volume 8, pp. 1433–1438.
31. Finn, C.; Levine, S.; Abbeel, P. Guided cost learning: Deep inverse optimal control via policy optimization. In Proceedings of the International Conference on Machine Learning, New York City, NY, USA, 19–24 June 2016; pp. 49–58.
32. Zhang, T.; McCarthy, Z.; Jowl, O.; Lee, D.; Chen, X.; Goldberg, K.; Abbeel, P. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In Proceedings of the IEEE International Conference on Robotics and Automation, Brisbane, Australia, 21–25 May 2018; pp. 1–8.
33. Finn, C.; Yu, T.; Zhang, T.; Abbeel, P.; Levine, S. One-Shot Visual Imitation Learning via Meta-Learning. In Proceedings of the Conference on Robot Learning, Mountain View, CA, USA, 13–15 November 2017; Volume 78, pp. 357–368.
34. Fang, B.; Jia, S.; Guo, D.; Xu, M.; Wen, S.; Sun, F. Survey of imitation learning for robotic manipulation. *Int. J. Intell. Robot. Appl.* **2019**, *3*, 362–369.
35. Ding, Y.; Florensa, C.; Abbeel, P.; Phielipp, M. Goal-conditioned imitation learning. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; pp. 15324–15335.
36. Gangwani, T.; Liu, Q.; Peng, J. Learning self-imitating diverse policies. *arXiv* **2018**, arXiv:1805.10309.
37. Wu, Y.H.; Charoenphakdee, N.; Bao, H.; Tangkaratt, V.; Sugiyama, M. Imitation Learning from Imperfect Demonstration. In Proceedings of the International Conference on Machine learning, Long Beach, CA, USA, 10–15 June 2019.
38. Tang, Y. Self-imitation learning via generalized lower bound q-learning. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 6–12 December 2020.
39. Guo, Y.; Choi, J.; Moczulski, M.; Bengio, S.; Norouzi, M.; Lee, H. Self-Imitation Learning via Trajectory-Conditioned Policy for Hard-Exploration Tasks. *arXiv* **2019**, arXiv:1907.10247.
40. Rauber, P.; Ummadisingu, A.; Mutz, F.; Schmidhuber, J. Hindsight policy gradients. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
41. Fang, M.; Zhou, C.; Shi, B.; Gong, B.; Xu, J.; Zhang, T. DHER: Hindsight Experience Replay for Dynamic Goals. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
42. Liu, N.; Lu, T.; Cai, Y.; Li, B.; Wang, S. Hindsight Generative Adversarial Imitation Learning. *arXiv* **2019**, arXiv:1903.07854.
43. Zhao, R.; Tresp, V. Energy-based hindsight experience prioritization. *arXiv* **2018**, arXiv:1810.01363.
44. Fang, M.; Zhou, T.; Du, Y.; Han, L.; Zhang, Z. Curriculum-guided hindsight experience replay. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; pp. 12623–12634.
45. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; MIT Press: Cambridge, MA, USA, 2018.
46. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.

47. Andrychowicz, O.M.; Baker, B.; Chociej, M.; Jozefowicz, R.; McGrew, B.; Pachocki, J.; Petron, A.; Plappert, M.; Powell, G.; Ray, A.; et al. Learning dexterous in-hand manipulation. *Int. J. Robot. Res.* **2020**, *39*, 3–20.
48. Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Dębiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; et al. Dota 2 with large scale deep reinforcement learning. *arXiv* **2019**, arXiv:1912.06680.

Publisher’s Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).