Improving sample efficiency of robot grasping

Guanang Su
Northeastern University
su.gu@northeastern.edu

Mingxi Jia
Northeastern University
jia.ming@northeastern.edu

***Abstract*—In goal-conditioned reinforcement learning problems, the sample efficiency is often a drawback that most of the explorations are not consider as very useful experience because they are failure episodes, which makes low sample efficiency. In this paper, we implemented a module of Hindsight Experience Replay (HER) in several goal-conditioned environments, to discover its utility of improving sample efficiency. Based on Deep Deterministic Policy Gradient (DDPG), the experiments showed that the HER module helps the agent learn much faster with more robustness. We then discussed about the limitation of HER and how hyper parameters effects its performance.**

***Index Terms*—reinforcement learning, robot grasping, deep learning, robotics, robot manipulation**

I. Introduction

Reinforcement learning (RL) is becoming a popular trend in the field of machine learning and robotics. RL provides a framework for learning state-depend decisions while interacting with the world (environment). A typical RL agent learns to find maximum expectations in every situation in order to eventually, get a reward which is often predefined in the environment. The pipeline is very general for different settings of tasks and often very handy for finding available paths for solving the problem, thanks to its inherent exploration and exploitation mechanism. Furthermore, the recent development of deep neural network also boost the reinforcement learning community, by its powerful function approximation ability. Several methods are introduced, varying from model-free RL to model-based RL. And, in model-free approaches, it varies from Q-learning to policy optimization. Some of which, such as DQN [1], Soft Actor-Critic (SAC) [2], Deep Deterministic Policy Gradient (DDPG) [3], are getting extremely great performance on different tasks like gaming, playing chess, etc.

However, for some tasks with large action/state space, e.g. grasping, pick and place, the exploration process becomes difficult as the search space is too large which causes so-called "curse of dimensionality". Especially in the goal-variant sparse rewarded environment, it takes too much time looking for one successful path to update, for example, the state q values as the goal is changing, leading to a even larger exploration space because we need state-action-goal pairs instead of state-action pairs. Most of the "failure experience" is wasted while we only utilize those "successful experience". As we reflect on human learning, human not only learn from success but as well from failures.

In order to make full use of failure experience, Andrychowicz, Marcin, et al [4], proposed Hindsight Experience Replay (HER) to improve the sample efficiency in goal-oriented tasks. By sampling sub-goals, episodes who fail to reach the original designated goal can be viewed as it successful achieve other goals on its exploration path. The specific mechanism of HER will be introduces in Section II.D. In this paper, we analysis the influence of HER on learning based on a deep policy-gradient method DDPG [3] which is commonly used on tasks which involve continuous state and action space.

In our experiment, we use the OpenAI Gym Robotics environments [5] to train and test different methods. Firstly, we use, FetchReach, FetchPush, FetchSlide, FetchPickAndPlace, these four robotics environments because these are goal-variant tasks with high dimensional state/action space, in which we have a 7 degree-of-freedom Arm with a two fingered parallel gripper mounted. Rewards are sparse that the agent only get a reward once achieving the goal.

Observation space:

- observation: robot state & position of objects
- desired goal: 3-D target position

- achieved goal: position of robots' end effector

Action space:

- 3-D movement (X, Y, Z, fixed θ)
- open/close state of the gripper

Fig. 1. FetchReach-v1.

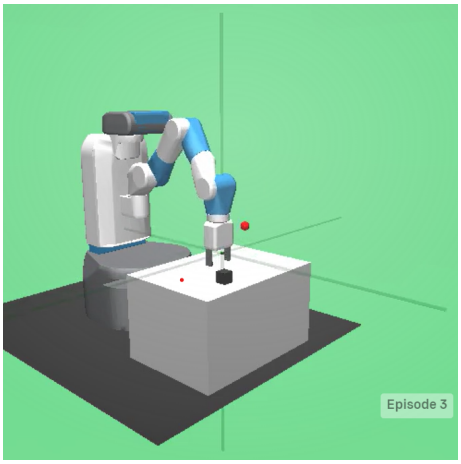

Fig. 2. FetchPickAndPlace-v1.

The OpenAI gym robotics inherits the Mujoco [5] simulation tool which gives real-time realistic physical simulation.

According to the final project requirements, this paper is organized as followed.

1) High-level understandings and motivations are addressed in abstract and introduction sections.
2) Technical problem statement is in the experiment section.

Fig. 3. FetchPush-v1.

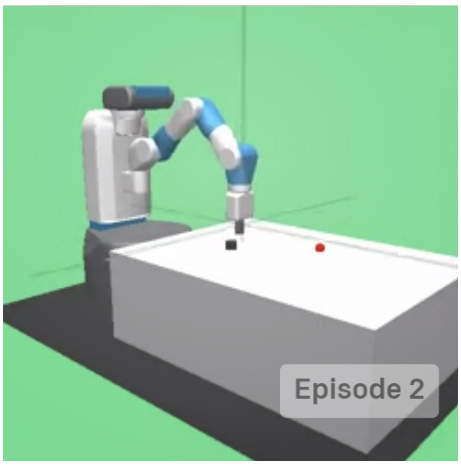

Fig. 4. FetchSlide-v1.

3) The simulation and data are introduces in the introduction section.
4) We describe Methods / algorithms in the background section.
5) References, Empirical results, and implementation details are in experiments section.
6) Analysis, discussions, and future direction are placed in the last two sections.

II. Background

A. Reinforcement Learning (RL)

Reinforcement Learning is a machine learning method based on rewarding desired behaviors and punishing undesired ones. The environment for RL is usually state in the form of Markov Decision Process (MDP) as the algorithms using dynamic programming techniques. An MDP consists of a

finite environment state S, In general, reinforcement learning learns a model from experience and use that to update modelled transitions for the value function. Some key elements that describe the basic elements of RL and relationship in between are list below and shown in Fig.5:

- environment: the physical world that agent operates
- state: (current) situation of agent
- action: activity of agent's next move
- reward: feedback of environment
- policy(or π): method to map agents' states and actions
- value: future reward that an agent could receive by taking an action in a specific state

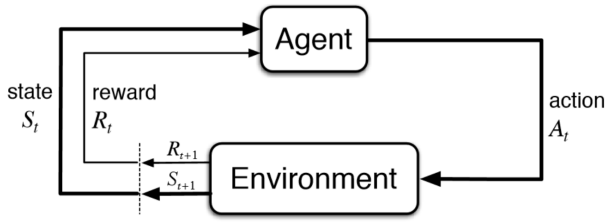

Fig. 5. Basic Element and Ideas of Reinforcement Learning.

As one of the basic machine learning methods alongside with supervised learning and unsupervised learning, reinforcement learning does not need labelled input/output and explicitly sub-optimal actions. Instead of get a balance between exploration and exploitation, reinforcement learning combines the advantage of supervised learning with the RL algorithms.

B. Multi-goal RL

The multi-goal RL has a very practical value for real-world robotics applications. Take a pick and place tasks as an example, if we only have a single goal during training, it means that, for different goals, we will have to train them separately which is time-consuming and as well impracticable knowing that the training is much slower than testing. We want our agent can find optimal path to different given goals after one-time off-line training, where typical RL agent only accept state as input without knowing the goal because it is defaultly considered a part of the environment. To solve this problem, Schaul et al. introduced Universal Value Function Approximators [9] by which we trained policy and value function which not only take a state but also a goal.

C. Hindsight Experience Replay (HER)

Dealing with sparse reward is one of the biggest challenge in reinforcement learning problem since it is hard for the agent to learn with reward in a large action space. The new technique HER presented by M Andrychowicz in 2017 [4]. This allows sample-efficient and learning from both sparse and binary reward. It can be combined with any off-policy algorithms, like DDPG which used in this project.

Unlike the the current model-free RL algorithms, human could learn from achieving an undesired goal as from desired outcomes. Therefore, HER could achieve multi-goal by treating each state of the system as separate goal. The algorithm behind HER is simple: we store some episodes from $s_0, s_1, ..., s_T$ in the replay buffer every transition $s_t \rightarrow s_{t+1}$ with both the original goal and a subset of other goals. Details shown in 15. HER may be seem as a form of a implicit circumstance as the goals used for replay naturally shift from simple goals to difficult ones. At the same time, HER does not need to control over the distribution of initial environment states.

D. Deep learning (DL)

Deep learning is in a rapid developing process recently. AlexNet [8] is the first Deep Neural Network (DNN) which outperforms any other methods at the time, showing the power of DNNs on image classification. Then, using the encoder-decoder architecture, DNNs are designed to be better on various fields, e.g. neural language processing, computer vision, and robotics.

Reinforcement learning also utilizes function approximation methods to extend its generalization ability. With the capability of non-linear approximation of neural networks, reinforcement learning can accommodate more complex environment. Many DL-based RL approaches even outperformed humans' performance on certain tasks. For discretized tasks, DQN [1] learns policy directly from pixels encoded by convolutional neural network, without manually designing features. For continuous tasks, other than discretizing action space, several approaches are introduced. DDPG [3] allows agents

to learn two locomotion tasks and Torcs (driving simulator). T Haarnoja et al. proposed SAC [4], training on a real-world legged robot which also learned gait which generalizes to unseen situations.

E. Deep Deterministic Policy Gradient (DDPG)

When think about reinforcement learning, Deep Q Network (DQN) and State-Action-Reward-State-Action (SARSA) are two commonly used model-free algorithms in use. These two algorithms can overcome some circumstances and derive a optimal policy from seen states. But DQN has limitation of it can only handle discrete, low-dimensional action spaces. Many real world problem have continuous and high-dimensional domains since it wish to maximize the action-value function and required iterative optimization process to finish that. While DDPG could solve those challenges. Combined with Deterministic Policy Gradient (DPG) [6] and DQN, the actor-critic approach with insight of DQN, a stable and robust off-policy with samples from replay buffer to minimize the correction between is formed, which called DDPG. [3]

In detail of introducing the network and algorithm, a graph is given in 18. In DDPG, two neural networks are maintained, a target policy (called $actor$) $\pi : S \rightarrow A$ and action-value function approximator (called $critic$) $Q : S \times A \rightarrow \mathbb{R}$. The critic's job is to approximate the actor's action value function Q^{π} and trained in a similar way as the Q-function but the target are computed using actions output by the actor. The actor is trained with mini-batch gradient descent on the loss that sampled from the replay buffer. Compared to other method, DDPG maintains a more stable, quicker shift between different environments and suitable for complex action space, which is suitable for the need of this project.

III. Experiments

A. Environments

The environments we used in this paper are generally introduced in section II.A, of their observation space, action space, and reward function. These four environments, FetchReach, FetchPush, FetchSlide, FetchPickAndPlace, are included in OpenAI gym robotics which is also based on Mujoco physical simulator. Four tasks has different difficulty level, in which the reaching task is the easiest; pushing and sliding are harder; the pick and place are the most difficult one. The difficulty level is determined mainly by the setting of the task:

- FetchReach: The gripper will try to reach a randomly chosen goal pose in each episode.
- FetchPush: The robot need to learn how to push a randomly positioned cube to a random goal.
- FetchSlide: Robot learns to push a cylinder once on a slippery surface to make the object stop at a random goal state.
- FetchPickAndPlace: The robot need to grasp a cube and take it to a randomly sampled goal state in a 3D space.

B. Learning with (DDPG)

DDPG is an actor-critic, model-free algorithm which can be used in continuous control problems. The first question is that can DDPG help agents learn in the mentioned four environments whatever the sample efficiency is, because sample efficiency matters only if the algorithm can converge in the given tasks. So, we train agents in our tasks using DDPG. Here are the results.

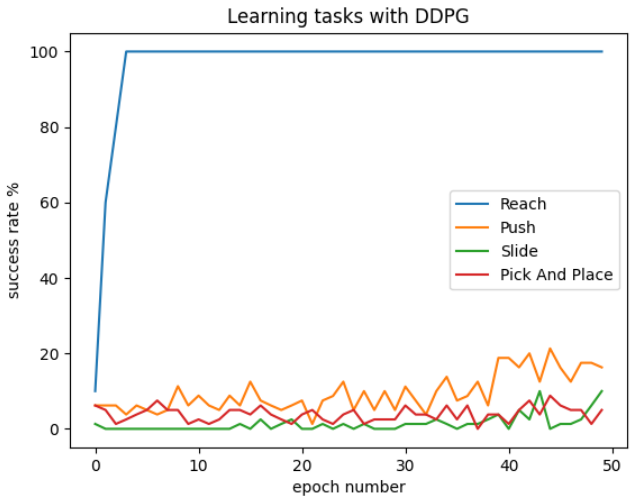

Fig. 6. How good agents learn in four given goal-oriented tasks. Except the FetchReach, the rest tasks are gradually learn something but in a very slow process. The success rate we are showing is tested in the evaluation.

C. Does HER improve the sample efficiency?

We then implement the HER algorithm as a plug-in module in the original DDPG so that we can

learn from failures also. As the HER declaims, the sample efficiency is improved by sampling sub-goals in every episodes no matter whether the agent eventually achieve the original goal. That is to say, with a efficient sampler, we expect to see that: (1) HER being a plug-in module, it, at least, will not impair the performance of DDPG; (2) HER will make the learning faster in the same amount of training time.

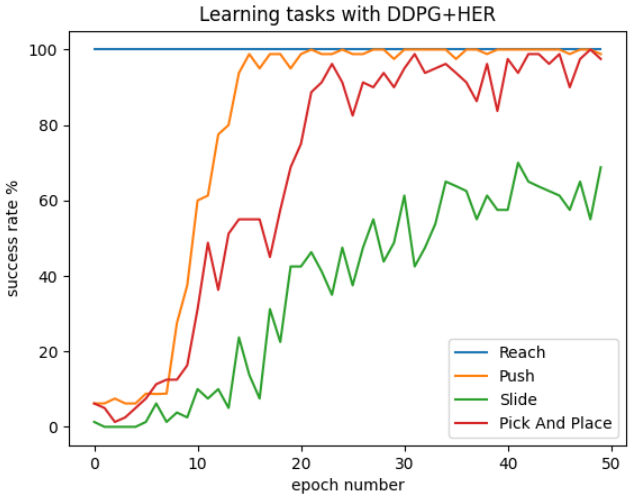

Fig. 7. All four tasks with DDPG+HER which dramatically boost the training process

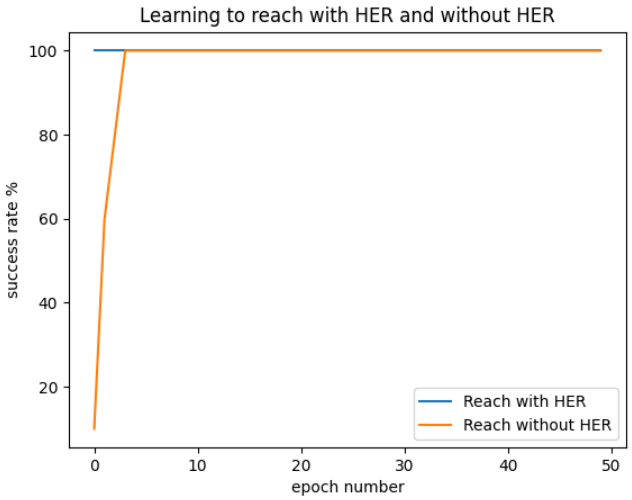

Fig. 8. Comparing FetchReach-v1 with HER (DDPG+HER) and without HER (only DDPG)

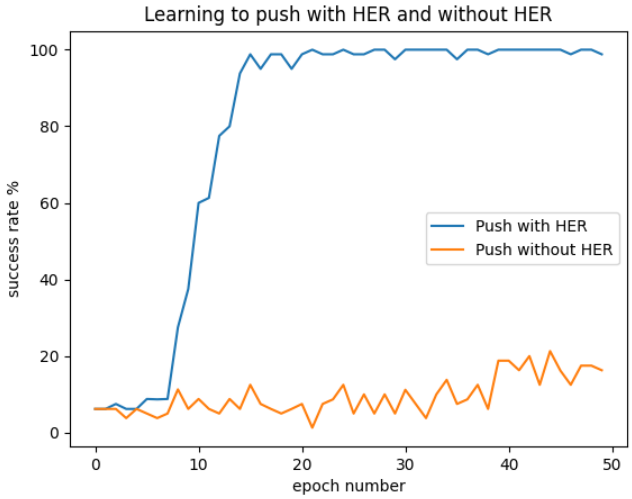

Fig. 9. Comparing FetchPush-v1 with HER (DDPG+HER) and without HER (only DDPG)

Fig. 10. Comparing FetchSlide-v1 with HER (DDPG+HER) and without HER (only DDPG)

The difficulties of tasks can be easily recognized on the evaluation curve shown above. Even though the reaching is only trained in single thread, the agent learned in a epoch. Surprisingly, sliding task is more difficult to learn than the pushing and pick & place task. A conjecture would be that, firstly, sliding is on a slippery surface and one-push limitation so that the arm has few opportunity to adjust the position of the object. Secondly, pick & place learning appears good because as we observe the actual trained model, the agent achieve nearly

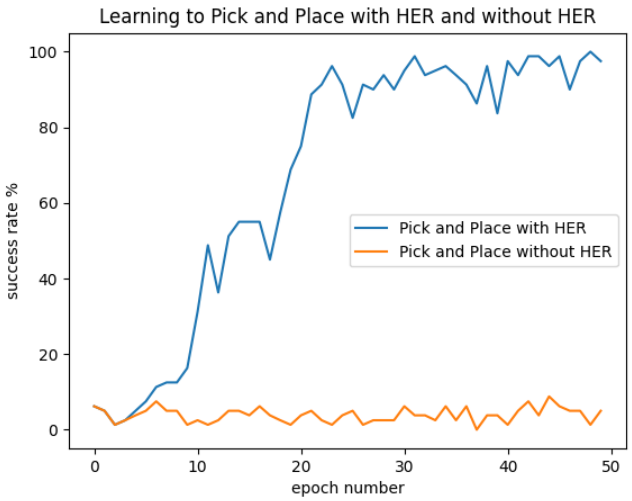

Fig. 11. Comparing FetchPick-v1 with HER (DDPG+HER) and without HER (only DDPG)

100% successful rate on simple goals, e.g. goals near the desk. For those hard goals (fewer) which is above the desk, the agent perform badly. So, if hard goals appears more, pick & place will probably show lower successful rate compared with sliding task.

Comparing with DDPG only, HER dramatically make the training process of hard tasks easier by improving the sample efficiency. Without using HER, push, slide, and pick & place barely learn nothing in the first epoch. Instead, by adding HER into DDPG, an obvious rising the learning curve shows that the idea of learning from failure is essential especially for tasks which have large action/state space with variant goals.

D. Implementation details

In our experiments, parallel training are deployed, except that FetchReach are non-parallel trained, because it is already fast to reach optimal policy in a single thread. Others are trained in eight threads. We uses 8-thread parallel training so that each epoch contains 8 episodes. Also, because we are using universal value function approximator, so we are finding the value of each state-action-goal pair instead of typical state-action pair. In order to do so, as the actor-critic model is used, we store state-action-goal pair into our buffer and the input dimension of the neural network is observation plus goal.

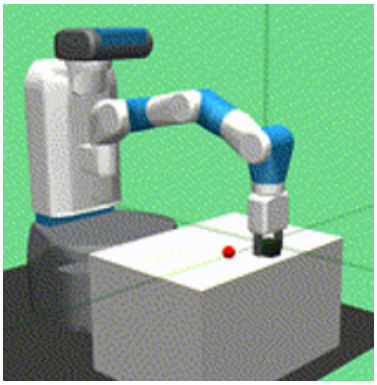

Fig. 12. Easy pick & place.

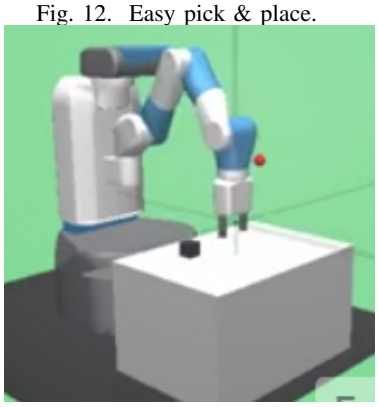

Fig. 13. Hard pick & place.

In addition, the HER module is implemented as a modified sampler in the replay buffer where the module set some modified goal for some certain pairs. We only implemented the ”future” strategy that we ”replay with k random states which come from the same episode as the transition being replayed and were observed after it” [4].

IV. Conclusion

These tasks are very different from the four-room domain task as we used in the class. The four-room is a single-goal environment that our agent only need to find a optimal path from the origin to a designated goal which keep constant in different episodes. In contrast, the environment we are using in this paper is goal-variant which means goals varies in episodes, which are randomly selected. The changing goal makes the training harder and requires high non-linear approximation ability of our function approximator.

To address this problem, HER make full use of failure episodes to enrich agent's experience of various goal. The approach is shown to be effective as

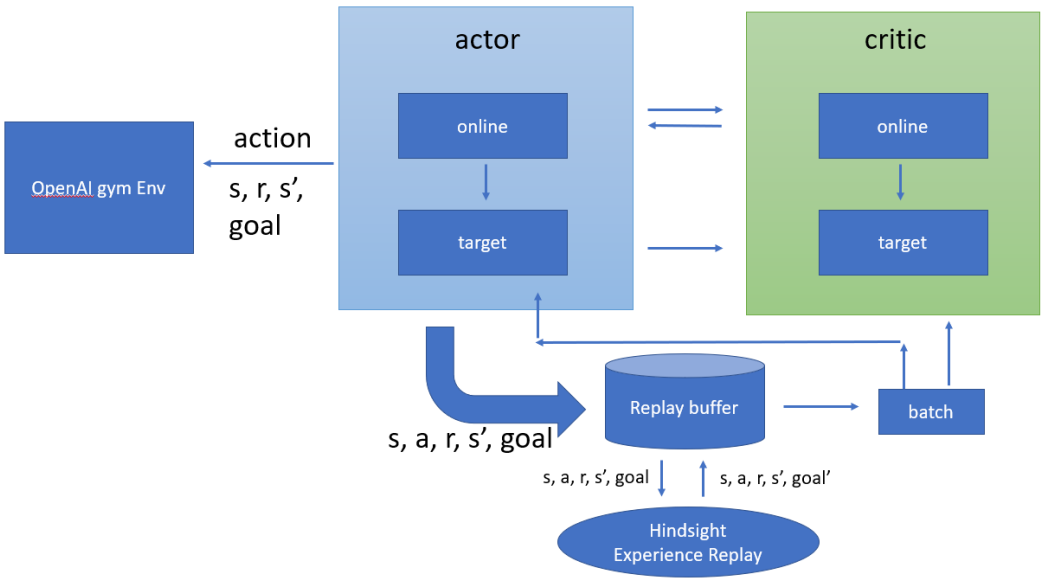

Fig. 14. HER+DDPG architecture

the experiment results imply. HER can dramatically improve the sample efficiency of multi-goal problem so that the training is much faster with robustness.

V. Future Diction

As mentioned, HER is limited that we need to know the current state coordinate and the goal which is hard to get in most of the real-world problems where states are represented as e.g. images. For example, we want a robot arm to grasp a object on the desk and we have a goal image which is pick and place the object to another certain place. HER cannot work in this situation because it need to access the true-state reward function. In the future, we should extend HER to image-based tasks.

A Nair et al. [10] proposed a visual RL with image goals where we measure the difference between images by comparing the distance of their latent vector which is encoded by convolutional neural networks. Based of this idea, the HER can be extended to adapt image-based problems in the future.

Acknowledgment

This paper is a final project outcome for CS5180 Reinforcement Learning and Sequential Decision lectured by Prof. Robert Platt. We are thankful for Prof. Platt's and TAs' (Xupeng Zhu, John Park) help, which teach us to comprehensively know the literature so that this project is possible. Here is our link for the code: https://github.com/roboticTeam/robotArm_grasping.git

References

[1] Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).
[2] Haarnoja, Tuomas, et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor." International conference on machine learning. PMLR, 2018.
[3] Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." arXiv preprint arXiv:1509.02971 (2015).
[4] Andrychowicz, Marcin, et al. "Hindsight experience replay." arXiv preprint arXiv:1707.01495 (2017).
[5] Plappert, Matthias, et al. "Multi-goal reinforcement learning: Challenging robotics environments and request for research." arXiv preprint arXiv:1802.09464 (2018).
[6] Silver, David, Lever, Guy, Heess, Nicolas, Degris, Thomas, Wierstra, Dann, and Riedmiller, Martin. "Deterministic policy gradient algorithms". ICML, 2014.
[7] Todorov, Emanuel, Tom Erez, and Yuval Tassa. "Mujoco: A physics engine for model-based control." 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2012.
[8] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems 25 (2012): 1097-1105.
[9] Schaul, Tom, et al. "Universal value function approximators." International conference on machine learning. PMLR, 2015.
[10] Nair, Ashvin, et al. "Visual reinforcement learning with imagined goals." arXiv preprint arXiv:1807.04742 (2018).

Appendix

Algorithm 1 Hindsight Experience Replay (HER)

Given:

- an off-policy RL algorithm $\mathbb{A}$, ▷ e.g. DQN, DDPG, NAF, SDQN
- a strategy $\mathbb{S}$ for sampling goals for replay, ▷ e.g. $\mathbb{S}(s_0, \ldots, s_T) = m(s_T)$
- a reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \to \mathbb{R}$. ▷ e.g. $r(s, a, g) = -[f_g(s) = 0]$

Initialize $\mathbb{A}$ ▷ e.g. initialize neural networks
Initialize replay buffer R
for episode = $1, M$ **do**
  Sample a goal g and an initial state s_0.
  for $t = 0, T-1$ **do**
    Sample an action a_t using the behavioral policy from $\mathbb{A}$:
      $a_t \leftarrow \pi_b(s_t||g)$ ▷ $||$ denotes concatenation
    Execute the action a_t and observe a new state s_{t+1}
  end for
  for $t = 0, T-1$ **do**
    $r_t := r(s_t, a_t, g)$
    Store the transition $(s_t||g,\ a_t,\ r_t,\ s_{t+1}||g)$ in R ▷ standard experience replay
    Sample a set of additional goals for replay $G := \mathbb{S}(\textbf{current episode})$
    for $g' \in G$ **do**
      $r' := r(s_t, a_t, g')$
      Store the transition $(s_t||g',\ a_t,\ r',\ s_{t+1}||g')$ in R ▷ HER
    end for
  end for
  for $t = 1, N$ **do**
    Sample a minibatch B from the replay buffer R
    Perform one step of optimization using $\mathbb{A}$ and minibatch B
  end for
end for

Fig. 15. HER algorithm

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

Fig. 16. DDPG algorithm

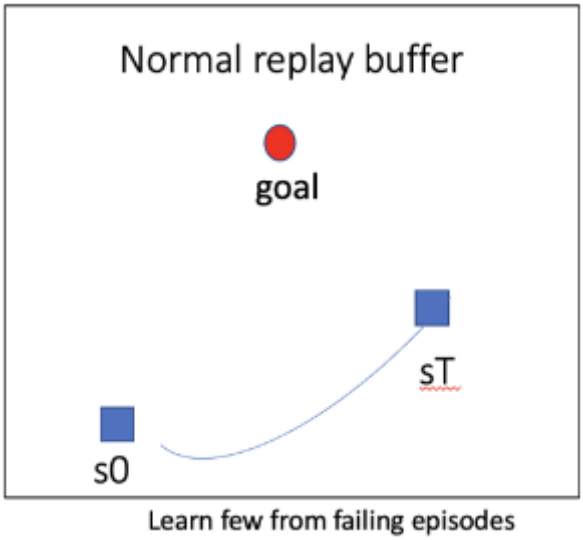

Fig. 17. Without HER, a typical experience buffer learn very less from failure episodes.

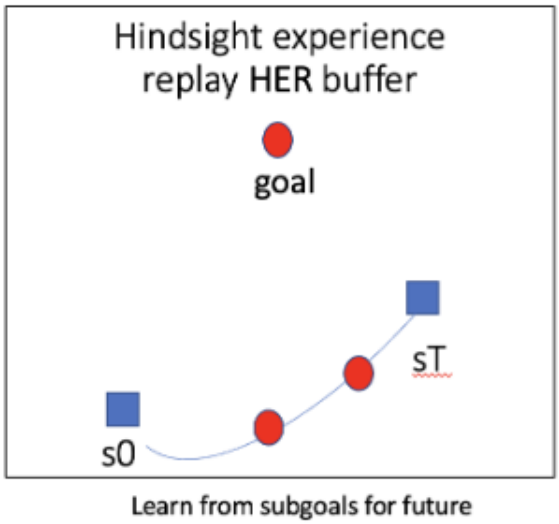

Fig. 18. With HER, by sampling subgoals from failure episodes (failure to achieve the original goal), the agent learn from failures.