Sept 30, 2025

TEXAS
The University of Texas at Austin

Hindsight Experience Replay(HER)

Rachel Yamamoto
The University of Texas at Austin

Engineering Reward Functions

- Rewards
  - Reflect task at hand
  - Shaped to help learning
- Reward engineering
  - Domain specific knowledge + RL knowledge
  - Know admissible behavior
- Real World Limitations
  - Costly and time-consuming to design and tune
  - Not scalable to many tasks or new environments
  - Infeasible when desired behavior is unknown or hard to specify
- Unintended or suboptimal behavior from poorly designed rewards

What if we could learn from unshaped rewards?

Rewards Used for Policy Optimization

Intermediate/Dense Awards

- Awarding for getting closer to the goal
- Frequent feedback, guiding step by step
- Ex. making progress around the track in car racing

$$r = \begin{cases} +1, & \text{good job!} \\ -1, & \text{you failed!} \end{cases}$$

Sparse and Binary Awards

- Vast majority of states give no informative reward signal
- Steps in between goal give little to 0 feedback
- Agent receives reward so infrequently
- Ex. win/loss rewards

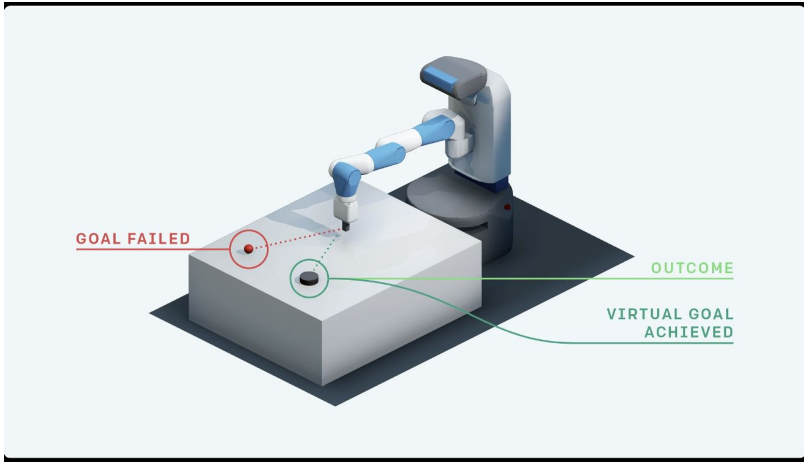
Sparse Reward Problem Consequences

- Inefficient search/ Slow learning
  - Unlikely to stumble upon rewarding state by chance
  - Long time to find reward signal
- Credit Assigning problem
  - Struggles to understand which sequence of actions led to failure or rare success
- Poor Exploration
  - Explore large region without finding reward + get stuck in local optima

Sparse Reward Solutions

- Reward shaping
  - Design rewards for intermediate steps
  - But…
    - Needs human expertise
    - Needs to be well designed
      - Otherwise: lead to suboptimal policies
- Hindsight Experience Replay

Hindsight Experience Replay (HER): Intuition



- A failed scenario for the target goal is a successful scenario for a different goal

- Paper Ex: You miss a goal in hockey, but what if the goal was placed where you hit the puck?

HER

- Replay each episode with a different goal than the one the agent was trying to achieve
  - Learning possible for sparse and binary rewards
  - Used with ANY off-policy RL algorithm
  - Can deal with multiple goal situations

Question: Any Off-policy RL Algorithm?

- HER needs to be trained on multiple goals.
- DQN is trained for a specific task or goal.
- How to extend this?

From One Goal to Multi-Goal

- Universal Value Function Approximators
  - DQN extension from one goal to multi-goal
  - Policy and value function take in s = state and g = goal
- Details
  - At the beginning of each episode, state-goal is sampled from p(s0, g)

$$\boxed{\pi : \mathcal{S} \times \mathcal{G} \rightarrow \mathcal{A}}$$

$$\boxed{r_t = r_g(s_t, a_t)}$$

$$\boxed{Q^{\pi}(s_t, a_t, g) = \mathbb{E}[R_t | s_t, a_t, g]}$$

Multi-goal RL

- Rewards depend on goals now
- One goal sparse rewards
  - Does this state achieve the goal?
- Multi-goal sparse rewards
  - How to check state satisfies particular goal?
    - Does state s satisfy goal g?
    - Predicate function

$$\boxed{r_t = r_g(s_t, a_t)}$$

Multi-goal RL: Predicate Function

Predicate function

$$\forall g \in \mathcal{G}, \quad f_g : \mathcal{S} \rightarrow \{0, 1\}$$
$$f_g(s) = 0$$
$$f_g(s) = 1$$

Multi-goal RL Sparse Reward

$$r_g(s, a) = -[f_g(s) = 0]$$

Question: How to relabel transitions with a new goal?

- Let’s create a mapping that maps states to the goal that is satisfied in that state
- Function answers the question: What goal would this state satisfy?

Mapping States to Goals

$$\boxed{m : \mathcal{S} \rightarrow \mathcal{G} \text{ s.t. } \forall_{s \in \mathcal{S}} f_{m(s)}(s) = 1}$$

=> mapping states to goals. g = m(s)

$$\boxed{\mathcal{G} = \mathcal{S} \text{ and } f_g(s) = [s = g]}$$

=> if this is the case, then the mapping is just the identity. m(s) = s

$$\boxed{\mathcal{G} = \mathbb{R} \text{ and } f_g((x, y)) = [x = g]}$$

$$\boxed{m((x, y)) = x}$$

=> extracting 1D goal from 2D state

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 $G := \mathbb{S}($**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

HER Algorithm

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 $G := \mathbb{S}($**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

- Run RL algorithm for M episodes
- For each episode, select a goal and an initial state

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 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 𝔸:
            a_t ← π_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 := 𝕊(current episode)
        for g' ∈ 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 𝔸 and minibatch B
    end for
end for
```

- Go through all steps in the episode and and interact with environment
- Choose an action and see result as a new state

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 $G := \mathbb{S}($**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

For each step in the episode:

=> storing transitions in the replay buffer

=> for each new goal, replace each transition with the new goal. Place in replay buffer

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 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 𝔸:
            a_t ← π_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 := 𝕊(current episode)
        for g' ∈ 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 𝔸 and minibatch B
    end for
end for
```

=> Train on a batches chosen from the replay buffer

Bit Flipping Example

Consider a bit-flipping environment with the state space $\mathcal{S} = \{0, 1\}^n$ and the action space $\mathcal{A} = \{0, 1, \ldots, n-1\}$ for some integer n in which executing the i-th action flips the i-th bit of the state.

$$r_g(s, a) = -[s \neq g]$$

Regular RL: Bit Flipping N = 3, T = 3, Target g = 111

Failure Episode

1. Initial state s0 = 000
   a. Take action flip bit 0 (a0)
2. s1= 100
   a. r(s0,a0) = -1
   b. Take action flip bit 1 (a1)
3. s2= 110
   a. r(s1,a1) = -1
   b. Take action flip bit 1 (a1)
4. s3= 100
   a. r(s2,a1) = -1

Successful Episode

1. Initial state s0 = 000
   a. Take action flip bit 0 (a0)
2. s1= 100
   a. r(s0, a0) = -1
   b. Take action flip bit 1 (a1)
3. s2= 110
   a. r(s1, a1) = -1
   b. Take action flip bit 2 (a2)
4. s3= 111
   a. r(s2,a2) = 0

Problems with Regular RL

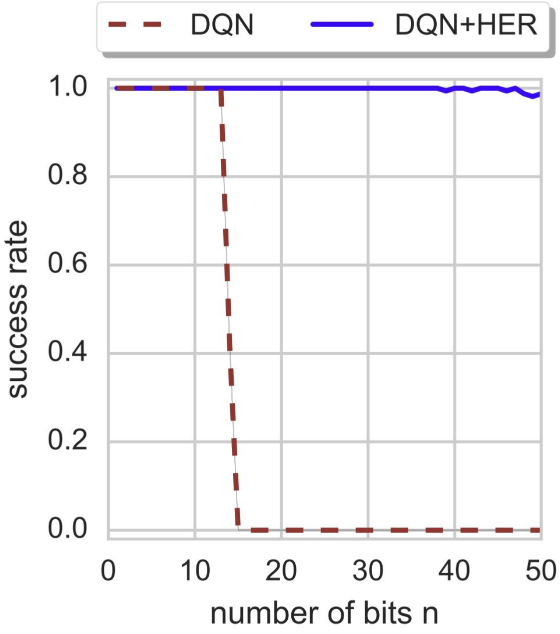
- But what if N = 40?
  - Large number of rewards will be -1 (might never experience something else)
- Reward shaping (not recommended)

$$r_g(s, a) = -||s - g||^2$$

Bit Flipping HER: N = 3, T= 3, Target g = 111

Failure Episode

1. Initial state s0 = 000
   a. Take action flip bit 0 (a0)
2. s1= 100
   a. r(s0,a0) = -1
   b. Take action flip bit 1 (a1)
3. s2= 110
   a. r(s1,a1) = -1
   b. Take action flip bit 1 (a1)
4. s3= 100
   a. r(s2,a1) = -1
5. Replace g with s3 in replay buffer



Other Experiments Set Up

- Simulate 7-DOF Fetch Robotics arm in MuJuCo physics engine

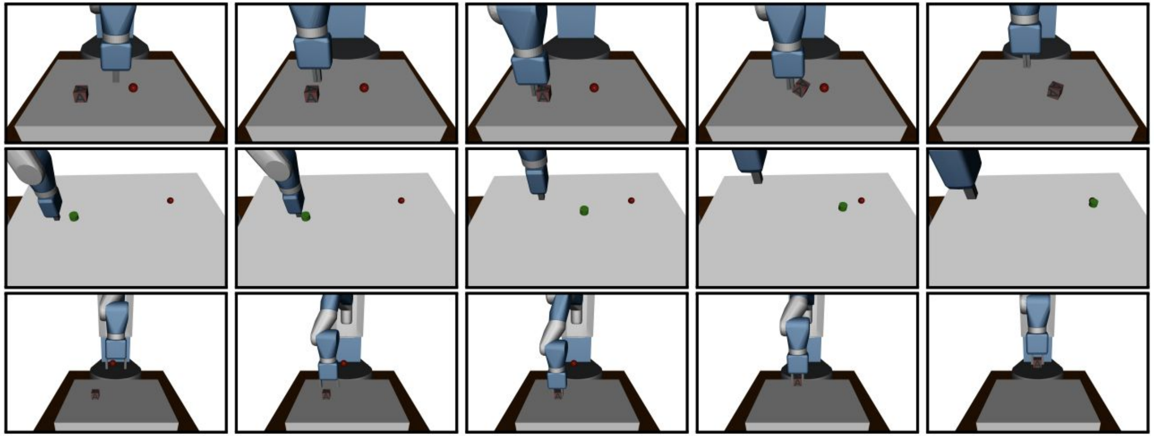


Figure 2: Different tasks: *pushing* (top row), *sliding* (middle row) and *pick-and-place* (bottom row). The red ball denotes the goal position.

Experiment SetUp

- States
  - Represented in simulation (robot angles, robot velocities of joints, object positions, object velocities)
- Goals
  - Desired position of object with some tolerance

$$m(s) = s_{\mathbf{object}}$$

$$r(s,a,g) = -[|g - s_{\mathbf{object}}| > \epsilon]$$

Experiment SetUp

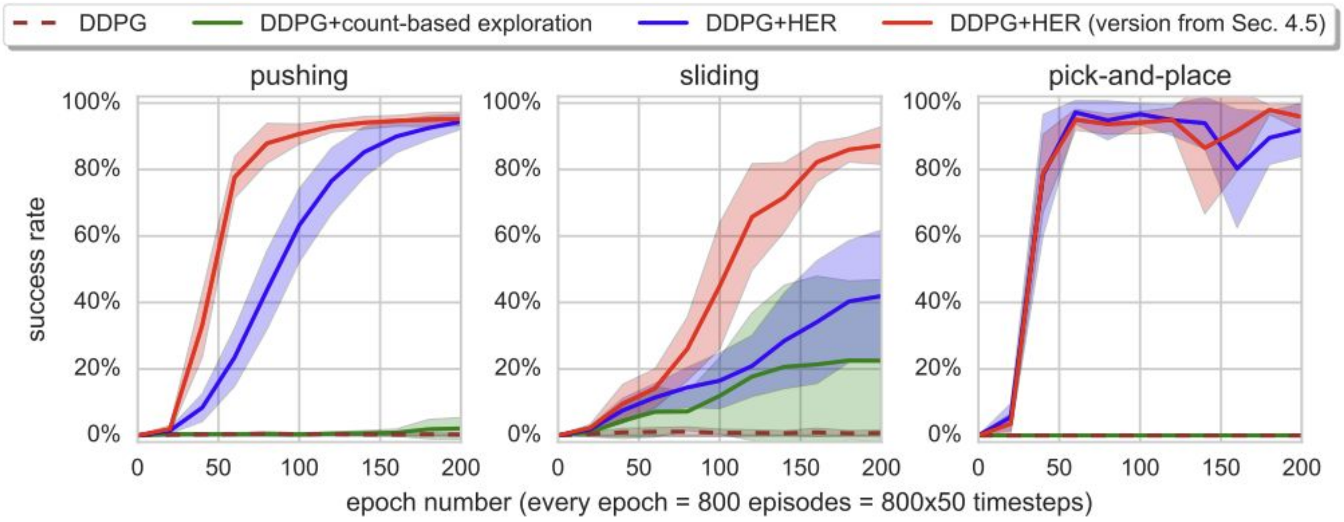
- Rewards
  - Sparse and binary rewards

$$\boxed{r(s,a,g) = -[f_g(s') = 0]}$$

- Strategy for sampling goals for replay
  - Goal corresponding to final state in episode

$$\boxed{\mathbb{S}(s_0,\ldots,s_T) = m(s_T).}$$

Results of 3 Tasks



HER One Goal Performance

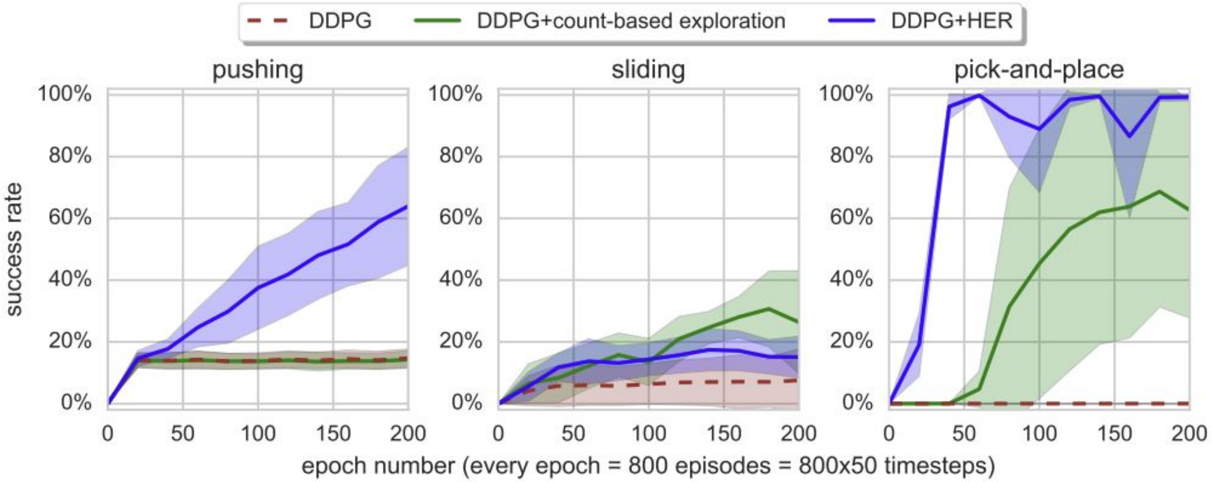


Figure 4: Learning curves for the single-goal case.

Reward Shaping Performance

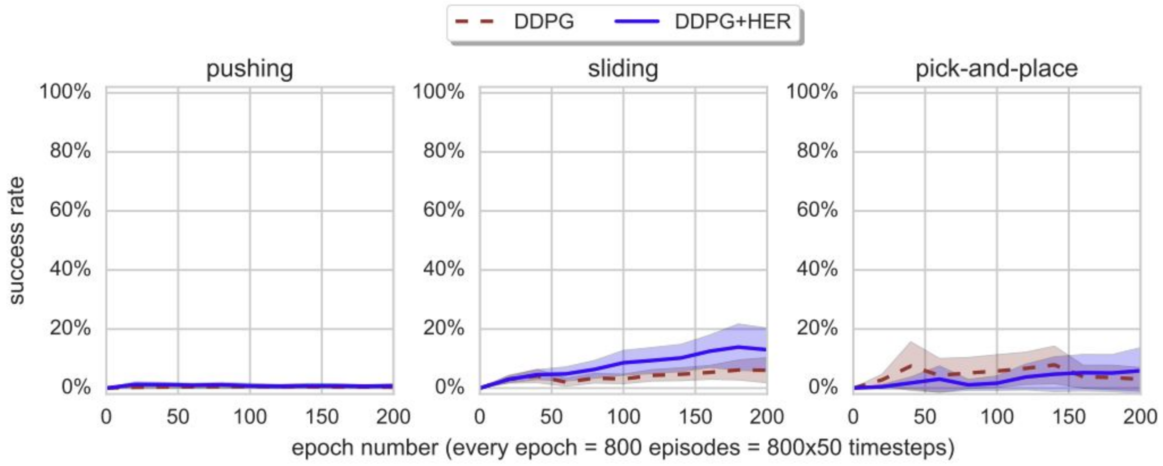


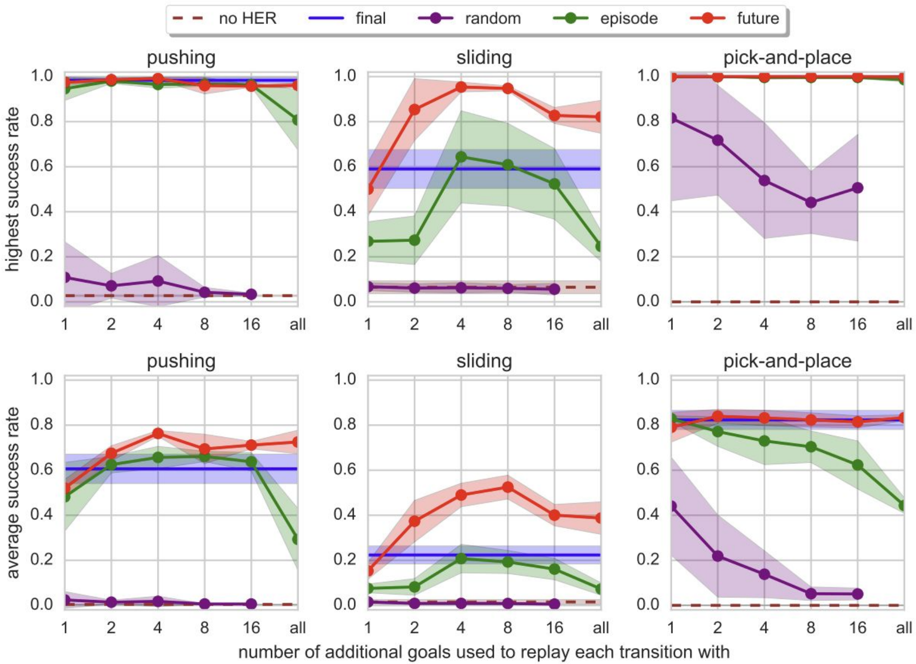
Figure 5: Learning curves for the shaped reward $r(s, a, g) = -|g - s'_{\mathbf{object}}|^2$ (it performed best among the shaped rewards we have tried). Both algorithms fail on all tasks.

Choosing Different Goals

- Final - goals corresponding to the final state of the episode

- `future` — replay with k random states which come from the same episode as the transition being replayed and were observed *after* it,
- `episode` — replay with k random states coming from the same episode as the transition being replayed,
- `random` — replay with k random states encountered so far in the whole training procedure.

Sampling Goals Differently



The End

Pictures from
https://openai.com/index/ingredients-for-robotics-research/

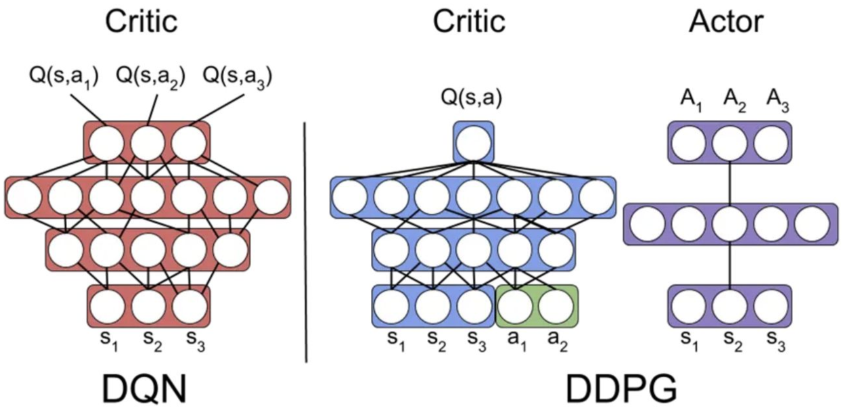
Continuous Action Spaces

- DQN supports finite action spaces
  - How to take the max of a continuous action space?
- DDPG
  - Q learning with continuous action spaces

Continuous Action Spaces - DDPG

- Actor NN (essentially the policy)
  - Input: states
  - Output: optimal action
- Critic NN
  - Used for calculating the Q value of the action from the actor NN
  - Input: state and action
  - Output: Q value

Continuous Action Spaces - DDPG



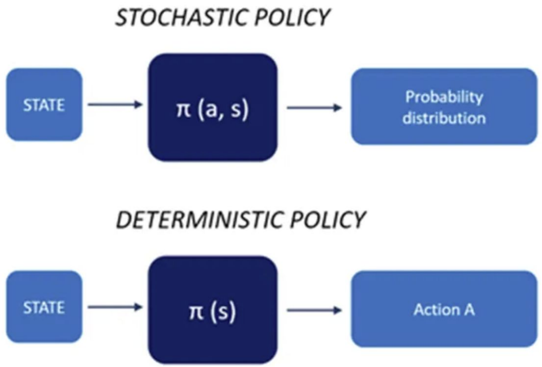
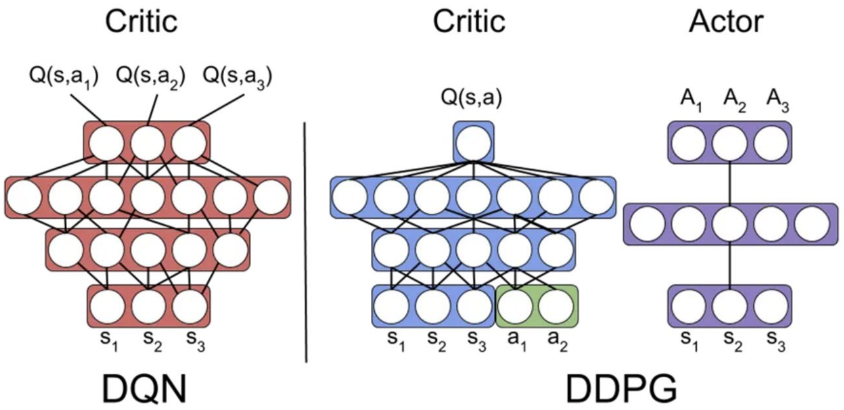
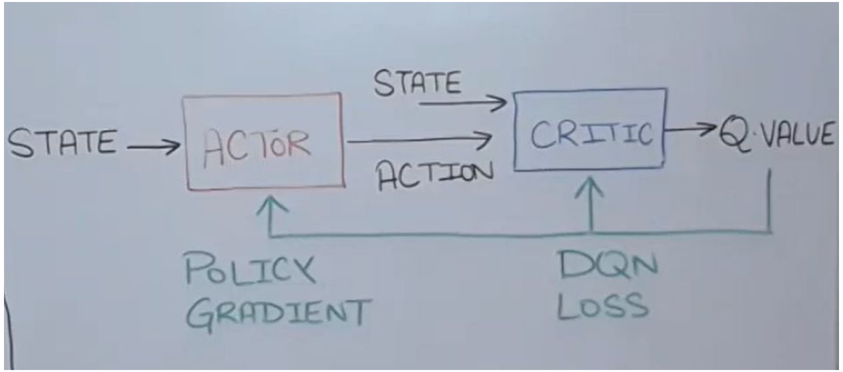


Image from this link

Continuous Action Spaces - DDPG





[Image from this link](#)
[Image2 from this link](#)