Dealing with the Sparse Reward Problem in the Bit-Flipping Environment

Xiaoyu Wen
Northwestern Polytechnical University
Xi'an, Shaanxi, P.R. China
wenxiaoyu@mail.nwpu.edu.cn

1 Problem Restatement

Consider a bit-flipping environment with a state space $\mathcal{S} = \{0, 1\}^n$ and an action space $\mathcal{A} = \{0, 1, ..., n-1\}$, where the integer n ranges from 1 to 50, and executing the i-th action flips the i-th bit of the state. For every episode we sample uniformly an initial state as well as a target state and the policy gets a reward of -1 as long as it is not in the target state, i.e. $r_g(s, a) = -[s \neq g]$. The objective of this project is to design and optimize a Deep Q-Network (DQN) to learn a efficient policy to transform an initial binary sequence into a target sequence though maximizing cumulative rewards.

2 Environment

We construct the bit-flipping experiment based on gym(https://github.com/openai/gym) and stable-baseline3(https://github.com/DLR-RM/stable-baselines3).

3 Methods

In this section, we evaluated the performance of different algorithm variants on this task.

3.1 Vanilla DQN

First of all, we design the following network architecture and training procedure based on CleanRL(https://github.com/vwxyzjn/cleanrl) to assess the efficacy of using vanilla DQN.

Network architecture: As shown in Figure 1, the Q-network has 1 hidden layer with 256 hidden units using ReLu activation function. We concatenate the current state and the target goal as input and use n dimensions as output for computing Q-targets. Here, we assume that only one action outputs at a time.

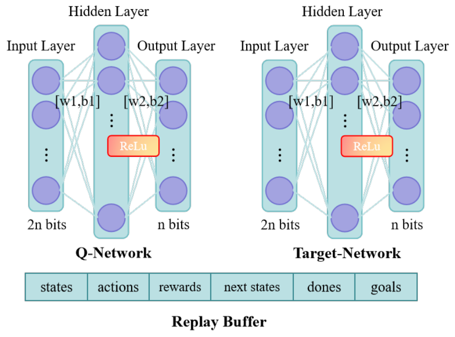

Figure 1. The setting of network architecture and buffer.

Training procedure: We train for 300k epochs. Each epoch consists of 1 iteration for data collection and 1 iteration for optimization on mini-batches of size 128, sampled uniformly from a replay buffer consisting of 10^6 transitions. We update the target networks after every iteration using a decay coefficient of 0.95. We use a learning rate of 0.001 with the Adam optimizer and a discount factor of $\gamma = 0.98$. The detailed hyperparameters of the baselines are listed below in Table 1.

Table 1. Hyperparameters used for vanilla DQN.

vanilla DQN hyperparameter	Value
Hidden layers	1(ReLU)
Hidden units	256(MLP)
Optimizer	Adam
Batch size	128
Replay buffer capacity	1e6
Discount factor γ	0.98
Target network update rate	0.95
Epsilon start	1.0
Epsilon end	0.05
Epsilon decay	0.001
Learning rate	0.001

Evaluation: We evaluated the performance of the Vanilla DQN algorithm by generating sequences of varying lengths, gradually increasing n from 1 to 50. As shown in Figure 2, the DQN is bound to fail in this environment for $n \geq 11$ because of the enormous state spaces. It's quite hard to get effective rewards other than -1. And with each additional dimension, the required epoch steps increase exponentially.

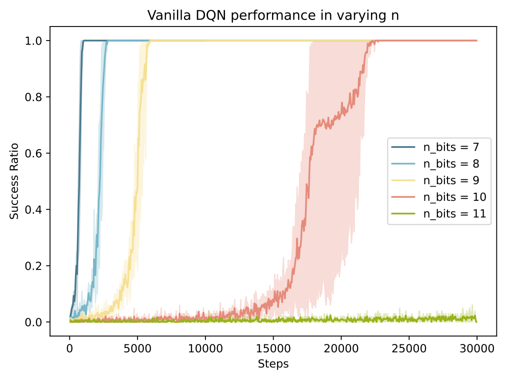

Figure 2. The performence of vanilla DQN in varying n.

Analysis: We observe that it works for small values of n, but doesn't work for larger values. So, what potential challenges are there? I think this problem can be approached from two situations:

- If the network outputs actions token by token, the enormous state space makes it challenging for the policy to gather effective training samples with positive reward. This is a typical sparse reward problem, which has been discussed in Hindsight Experience Replay [1]. The standard solution to this problem would be to use a shaped reward function which is more informative and guides the agent towards the goal. We investigate the results of reward shaping experimentally in Section 3.2. And another approach is to use Hindsight Experience Replay, a method based on retrospective goal adjustment that can effectively increase the proportion of positive rewards. We further explore the application of Hindsight Experience Replay (HER), which enhances learning efficiency by retrospectively assigning goals to transitions, thereby increasing the proportion of successful experiences in Section 3.3.
- If the network outputs multiple actions at once, we can use multiple network architectures to divide the state space, breaking down the large problem into multi-step subproblems. For instance, for $n = 50$, we can divide the state space into 5×10. We can select 5 same networks and each network handles 10 bits of the state to solve the problem. However, this approach requires manually selecting an appropriate way to split the state space dimensions, which lacks elegance and general applicability.

3.2 DQN with reward shaping

In this section, we modify the reward function to $r_g(s,a) = -||s-g||^2$ and validate the effectiveness of this approach. As shown in Figure 3, we observe that reward shaping can effectively improve the performance of vanilla DQN, with the success rate increasing rapidly. However, it's still bound to fail in this environment for $n \geq 34$.

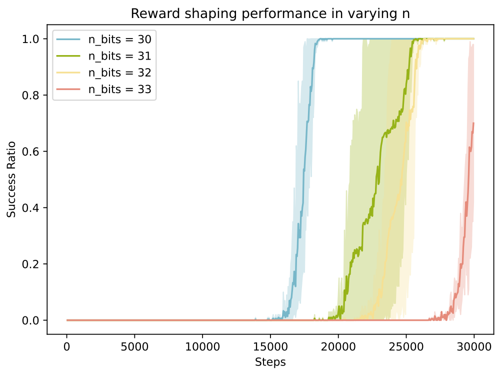

Figure 3. The performence of reward shaping in varying n.

3.3 DQN with hindsight experience replay

In this section, we first briefly introduce the main idea of Hindsight Experience Replay (HER) and several experience replay strategies including final, future, episode and random. And then we validate the performance of different strategies.

Main ideas: After the agent completes an episode, it retrospectively acts as if it was trying to achieve a different goal—specifically, one that it actually reached during the episode, even if it wasn't the original goal. The goal in transition *(state, action, reward, next state, **goal**)* is then replayed with this newly achieved state treated as the original goal.

Algorithm 1 Hindsight Experience Replay (HER)

```
1:  Initialize replay buffer $R$
2:  Initialize neural networks $Q$ and $Q'$
3:  for episode = 1 to $M$ do
4:      $s_0, goal \leftarrow$ env.reset()
5:      $episode \leftarrow []$
6:      $done \leftarrow False$
7:      while $done = False$ do
8:          $inputs \leftarrow$ concatenate$(s_t, g)$
9:          $a_t \leftarrow$ choose_action$(inputs, \epsilon)$
10:         $s_{t+1}, r_t, done \leftarrow$ env.step$(a_t)$
11:         $R \leftarrow R \cup \{(s_t, a_t, r_t, s_{t+1}, done, g)\}$
12:         $episode \leftarrow episode \cup \{(s_t, a_t, r_t, s_{t+1}, done)\}$
13:         $s_t \leftarrow s_{t+1}$
14:     end while
15:     for each transition in episode do
16:         Sample new goals $G \leftarrow \mathcal{S}$(current episode)
17:         for each $g' \in G$ do
18:             $r' \leftarrow r(s_t, a_t, g')$
19:             $R \leftarrow R \cup \{(s_t, a_t, r', s_{t+1}, done, g')\}$
20:         end for
21:     end for
22:     $B \leftarrow$ sample minibatch from $R$
23:     Perform one step of optimization using $Q$ and $B$
24:     Update $Q'$ using target network update
25: end for
```

Goal selection strategies:

- In the **final** goal strategy, we use the final state of the episode as the new goal.
- In the **future** goal strategy, we sample k random states from the episode that occur after the current transition.
- In the **episode** goal strategy, we sample k random states from the entire episode, not just future states.
- In the **random** goal strategy, we sample k random states from the entire set of states encountered during training. We will need to maintain a global buffer of states for this purpose.

Regarding the parameter k, we follow the original paper and adopt a default value of $k = 4$.

Ablation: First of all, we evaluated the performance of different strategies when $n = \{35, 40\}$. As shown in Figure 4, the future strategy demonstrate superior performance.

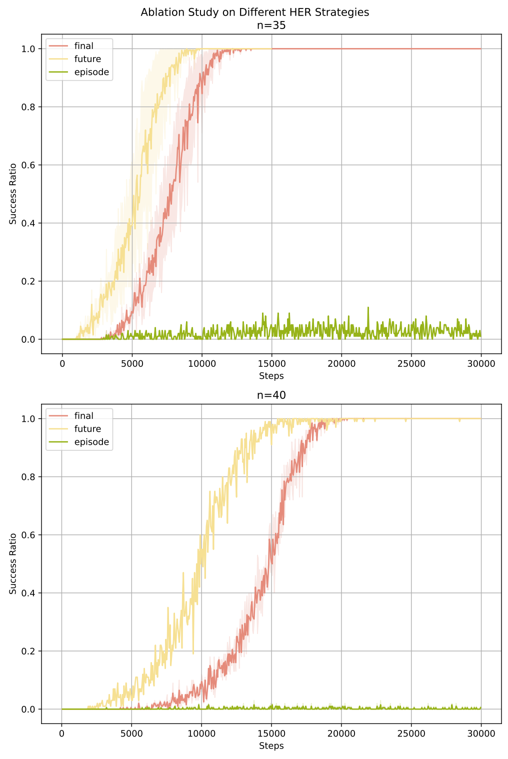

Figure 4. Ablation Study on Different HER Strategies.

Evaluation: We only selected the future strategy to validate HER's effectiveness in this environment by generating sequences of varying lengths gradually increasing n from 1 to 50. The result is presented below in Figure 6. However, for $n = 50$, HER needs more epoch steps for optimization because of enormous state space. So we presented the performance of HER with reward shaping compared with only HER. The result is presented below in Figure 5.

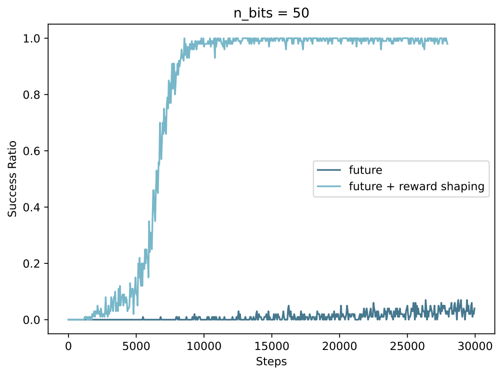

Figure 5. The performance of $n = 50$

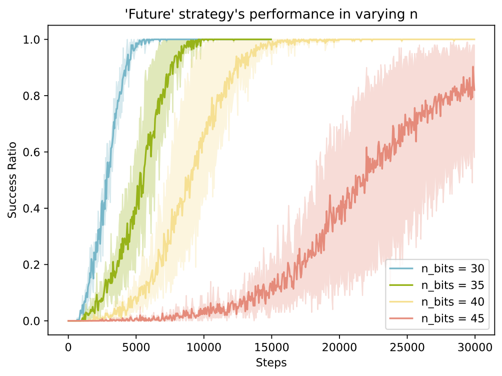

Figure 6. The performance of future strategy in varying n.

4 Conclusion

This project explores the use of Deep Q-Networks (DQN) in the Bit Flipping environment, aiming to address the challenge of sparse rewards as sequence lengths increase. We implemented reward shaping and Hindsight Experience Replay to enhance learning efficiency by allowing the agent to reinterpret failed experiences as successes with different goals. HER demonstrated significant improvements in learning efficiency compared to vanilla DQN and reward shaping, particularly for longer sequences.

References

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight experience replay. *Advances in neural information processing systems* 30 (2017).