REPRODUCING: UCB EXPLORATION VIA Q-ENSEMBLES

Zhaojie Gong, Yifan Jiang & Hengjia Zhang
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI, USA
{gzjmail, yifjiang, hengjia}@umich.edu

ABSTRACT

We reproduce the experiments in the paper UCB EXPLORATION VIA Q-ENSEMBLES and verify the main conclusions. The paper did one experiment on 49 Atari games and we attempted to replicate the results on one game (Up n Down). In addition, we also evaluate the models on a simpler environment CartPole. The authors of the original paper did not provide the code for each of the models. We implement the baseline model, Double DQN, as well as one of the proposed models, UCB Exploration. Our experiments show that the original paper is reproducible and their results are robust.

1 INTRODUCTION

One of the aims of deep reinforcement learning is to seek to learn mappings from high-dimensional observations to actions. One leading techniques that has been used successfully is Deep Q-learning (van Hasselt et al. (2015)). However, the challenge of this method remains like improving sample efficiency and ensuring convergence to high quality solutions.

The original paper proposes a Q-ensemble approach to deep Q-learning and UCB exploration strategy. They extend the intuition of UCB algorithms to the RL settings. THey use the outputs of a set of value functions and construct a UCB by adding the empirical standard deviation to empirical mean and choosing the action that maximizes UCB:

$$\text{action} = \underset{a}{\text{argmax}}\{\widetilde{\mu}(s_t, a) + \lambda \cdot \widetilde{\sigma}(s_t, a)\}$$

where λ is a hyper-parameter, while $\widetilde{\mu}(s_t, a)$ and $\widetilde{\sigma}(s_t, a)$ is the empirical mean and empirical standard deviation of $\{Q_k(s_t, a)\}_{k=1}^{K}$, respectively.

In this paper, we implement Double DQN and UCB exploration. Then we conduct experiments on 2 environments (1 Atari game UpNDown and CartPole) and compare the results of different methods.

2 METHODOLOGY

In this section, we propose our methods for reproducing the results in the original paper. We summarize the main conclusions drawn as follows:

1. Ensemble Voting outperforms Double DQN and bootstrapped DQN.
2. UCB Exploration outperforms Ensemble Voting.
3. Ensemble Voting and UCB Exploration both outperform count-based exploration method of Bellemare et al. (2016).

Due to the limit on time and computing resources, we focus on UCB Exploration and verify that it outperforms the Double DQN baseline. We train the two models on a simple environment as well as one Atari game selected from 49 games in the original paper.

2.1 BASELINE MODEL

The baseline model in the original paper is Double DQN. We implement it upon OpenAI baseline from Dhariwal et al. (2017) with Adam optimizer (Kingma & Ba, 2014). In order to reduce training time and the computing resources needed, the learning rate of the model is $1 * 10^{-4}$, which is larger than the one used in the original paper. Moreover, for the same reason, we set the number of $\{Q_k\}$ functions as $K = 5$ and the replay buffer size to be 10000. Other hyperparameters are the same as those listed in Table 1 of the original paper. Table 1 in Appendix A tabulates all the hyperparameters.

2.2 PROPOSED MODEL

The proposed model in the original paper is UCB Exploration based on Bootstrapped DQN. To make our reproduction comparable to the original paper, we implement it upon OpenAI baseline from Dhariwal et al. (2017) and make it exactly the same as Algorithm 2 as in the original paper. Same as the original paper, we fix the hyperparameters for Double DQN and UCB Exploration.

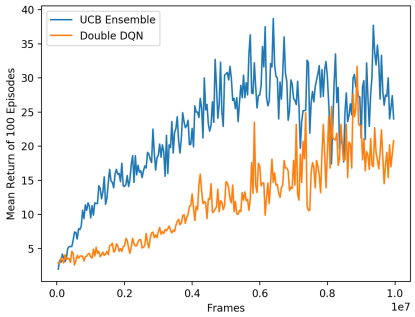
2.3 ENVIRONMENT

We evaluate the performance of the two models on two environments. The first environment is the Atari game UpNDown to compare with the original paper. We build the environment upon UpNDownNoFrameskip-v0 in OpenAI Gym from Brockman et al. (2016). We adapt the environment to repeating each action for four frames, which is the method used in the original paper.

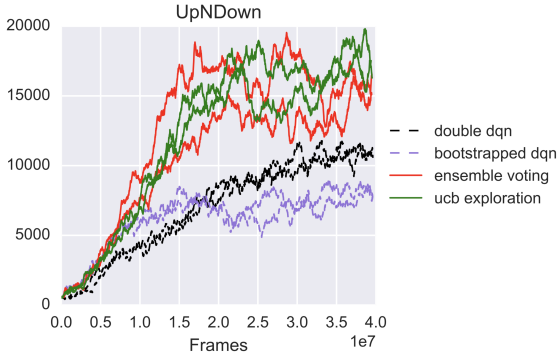
To further evaluate the robustness of the original paper's conclusion, we also train the two models on CartPole-v0 in OpenAI Gym from Brockman et al. (2016), which is a simpler environment than Atari games and is not included in the original paper. The input of the environment is a one-dimensional vector describing the status of the pole instead of an image, so we remove the CNN part of DQN for this.

3 RESULTS

We first evaluate Double DQN and UCB Exploration on UpNDown. Due to the time limit, we train the models for 10^7 frames, which is smaller compared with $4 * 10^7$ frames in the original paper. The value of λ is not provided in the original paper, and we set it to be 1. To compensate for this, we set the same initial value of the learning rate as in the original paper but do not decrease it over the training periods.



(a) Our result of the mean return of UCB exploration and Double DQN



(b) Original paper's result of the mean return of UCB exploration and Double DQN

Figure 1: Comparison of our result and original paper result

The learning curve in Figure 1a of our results is similar to that in Figure 1b. The scale of the reward is different because we probably use different versions of the environment. Our result shows that UCB exploration outperforms Double DQN on UpNDown, which supports the conclusion of the original paper. The UCB exploration method may facilitate efficient exploration in DQN, as the

original paper stated. Although the original paper does not provide the code, the hyperparameters provided in Appendix A make the replication much easier. However, the value of λ is not offered and the original paper does not interpret the meaning of the "learning curve" in their Figure 2. From their Table 2, we guess it represents the mean return in a window of 100 consecutive episodes.

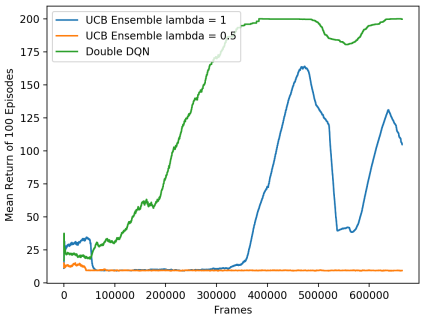


Figure 2: Learning curve of Double DQN on CartPole

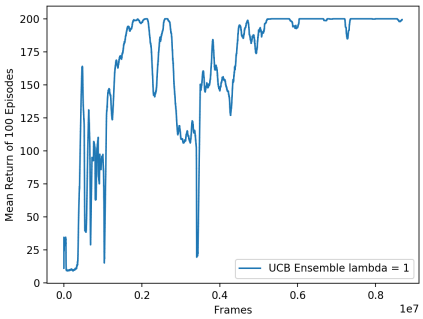


Figure 3: Learning curve of Double DQN on CartPole

Then we train both models on CartPole, a simpler environment not included in the original paper, to see whether the conclusion is robust across environments. The results for this environment are presented in Figure 2 and Figure 3. Double DQN outperforms UCB ensembles with $\lambda = 1$ and $\lambda = 0.5$. Figure 3 shows that the UCB ensemble needs more time to converge to the maximal reward. One possible reason is that the maximization bias is more significant in this environment, which damages the performance of UCB ensemble but does not affect Double DQN a lot.

The UCB ensemble with $\lambda = 1$ has better performance than $\lambda = 0.5$. This reflects the effect of λ to control the degree of exploration. The model with the smaller λ does less exploration and needs more time to achieve a better reward.

4 Conclusion

We successfully reproduce the original paper by reimplementing the proposed algorithms, comparing our results with those in the original paper and verifying one of their main conclusions. Our experiments show that the results in the original paper are reproducible and robust. In addition to simple replication, we also conduct an experiment on a different environment not included in the original paper to further evaluate the robustness of the main conclusion. The hyperparameter table provided in the original paper greatly helps the reproduction and improves the soundness of the paper.

REFERENCES

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 1471–1479. 2016.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Prafulla Dhariwal, Christopher Hesse, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Openai baselines. `https://github.com/openai/baselines`, 2017.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *ArXiv e-prints*, December 2014.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461, 2015.

A HYPERPARAMETER

We tabulate the hyperparameters in our well-tuned implementation of double DQN in Table 1:

hyperparameter	value	descriptions
total training frame	10 million	Length of training for each game.
minibatch size	32	Size of minibatch samples for each parameter update.
replay buffer size	10000	The number of most recent frames stored in replay buffer.
agent history length	4	The number of most recent frames concatenated as input to the Q network. Total number of iterations = total training frames / agent history length.
target network update frequency	10000	The frequency of updating target network, in the number of parameter updates.
discount factor	0.99	Discount factor for Q value.
action repeat	4	Repeat each action selected by the agent this many times. A value of 4 means the agent sees every 4th frame.
update frequency	4	The number of actions between successive parameter updates.
optimizer	Adam	Optimizer for parameter updates.
β_1	0.9	Adam optimizer parameter.
β_2	0.99	Adam optimizer parameter.
ϵ	10^{-4}	Adam optimizer parameter.
learning rate	10^{-4}	Learning rate for Adam optimizer
exploration schedule	$\begin{cases} Interp(1, 0.1) & t < 10^6 \\ Interp(0.1, 0.01) & otherwise \\ 0.01 & t > 5 \times 10^6 \end{cases}$	Probability of random action in ϵ-greedy exploration, as a function of the iteration t .
replay start size	50000	Number of uniform random actions taken before learning starts.

Table 1: Double DQN hyperparameters. $Interp(\cdot, \cdot)$ is linear interpolation between two values.