ELEN 6885 Project Report
Comparison of double-DQN and averaged-DDQN in Atari games

Jianhuan Zeng [1] **Xiaoning Wan** [1] **Yuwei Zhao** [1]

Abstract

To find an optimal algorithm for Atari games, we compare different algorithms, which include DQN, Double-DQN, Averaged-DQN and ADDQN. ADDQN is an algorithm combining Averaged-DQN with Double Q-learning. Specifically, we compare overestimations, performances and calculation cost of Atari Games in different algorithms by finding the value function estimation, average reward per episode and time cost. We find that both Averaged-DDQN and DDQN reduce overestimation and improve performance, but Averaged-DDQN also minimizing loss while DDQN increase loss in the opposite.

1. Introduction

Reinforcement learning (RL) is a distinct field of machine learning which seeks to learn optimal policy for sequential decision problems. Q-learning, as one of the most popular RL algorithms, could solve Markov Decision Processes (MDPs) optimally. However, sometimes, Q-learning trended to overestimate action values as a maximization step was included in this algorithm. To avoid this overestimation, double Q-learning and later double deep Q-network (DDQN) was proposed by researchers to achieve a lower variance and a higher stability (Van Hasselt et al., 2016).

Double-DQN algorithm provides an approach which reduces the chance of over optimistic, that is to prevent the overfitting from happening. However, as it uses the previous one step Q value for training, it suffers a same problem as DQN, the variance of Q value that makes the training process unstable.

Recently, Averaged-DQN come out to reduce approximation error (Anschel et al., 2017). It uses the K formerly learned Q-values to estimate the current action-value. ADQN was declared to able to be integrated with other DQN or DDQN simply. By using the average Q value of previous K steps for training, one can reduce the variance introduced by Q, and thus, stabilizes the whole training process.

Though there were comparison between DQN and averaged DQN, no detailed comparison of DDQN and averaged DDQN was presented. Therefore, in this project, we hope to explore the effectiveness of DDQN and averaged DDQN by using Atari Games, a typical RL model.

2. Background

This project is a part of study of Reinforcement Learning, based on Q-learning, Deep Q Networks(DQN) and Averaged DQN.

2.1. Reinforcement Learning

A general reinforcement learning framework is taken into consideration for this project (Sutton and Barto, 1998). It consists of an agent which interacts with the environment. The interaction is divided into discrete time steps that is defined as t = 0, 1, 2 At each time point t, the agent is observed at state s and several actions a could be chosen, which could lead to different rewards R. The aim of this RL model is to find a policy so that a maximum cumulative reward could be achieved.

There are many approaches to reach this goal. For example, by updating the state-value function Q (s, a), then get the policy which could attain the maximum Q(s,a). The maximum Q is defined as optimal state-action value function. Besides, we can update the policy directly. In this project. We focus on updating the state-action value function and then find the optimal policy.

2.2. Q-learning

The Q-learning algorithm is of great importance for RL algorithm as a temporal difference learning algorithm to solve the MDP model. The standard Q-learning algorithm updates the Q(s,a) value after action taken in state. The up-

*Equal contribution [1]Columbia University, New York, USA. Correspondence to: Jianhuan Zeng <jz2883@columbia.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

date algorithm is that the updated Q is equal to old Q plus the product of step size and TD-error. TD error is equal to return minus old Q; the return of Q-learning equal to reward plus the product of gamma and Q value for the best next state that is the largest among all possible next state.

However, in practice, due to the use of maximum operator, a positive bias could be produced, which is known as overestimation (Hasselt and Hado, 2010). This could influence the performance of Q-learning. Hence, the double q-learning and later double deep Q networks was proposed by researchers to overcome such drawbacks.

2.3. Deep Q Networks (DQN)

In 2013, researchers Mnih et al. proposed a DQN algorithm for supervised learning problems. DQN is based on state-value function update with experience replay and fixed target network. Convolution neural network is used for Q value evaluation, with a parameter, theta. The agent's experiences are stored at every time-step in a dataset D. In the learning process, for each batch of experiences, the Q-learning update is performed using the loss function.

Algorithm 1 DQN

Initialize $Q(s, a; \theta)$ with random weights θ_0
Initialize Experience Replay (ER) buffer $\mathcal{B}$
Initialize exploration procedure *Explore*$(\cdot)$
for $i = 1, 2, \ldots, N$ **do**
  $y^i_{s,a} = \mathbb{E}_{\mathcal{B}}\left[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) | \, s, a\right]$
  $\theta_i \approx \text{argmin}_\theta \, \mathbb{E}_{\mathcal{B}}\left[(y^i_{s,a} - Q(s, a; \theta))^2\right]$
  Explore$(\cdot)$, update $\mathcal{B}$
end for
output $Q^{\text{DQN}}(s, a; \theta_N)$

Figure 1. DQN Algorithm. theta is weight, B is Experience Replay buffer and Explore(dot)is exploration procedure.

2.4. Averaged DQN

To reduce approximation error, Averaged-DQN, a simple extension of DQN algorithm, uses the K formerly learned Q-values to generate the new action-value (Anschel, 2017). Researchers claims that Averaged-DQN could be easily integrated with other DQN or DDQN. Compared to DQN, the computational effort is more forward passes through a Q-network while minimizing the DQN loss.

Algorithm 2 Averaged DQN

Initialize $Q(s, a; \theta)$ with random weights θ_0
Initialize Experience Replay (ER) buffer $\mathcal{B}$
Initialize exploration procedure *Explore*$(\cdot)$
for $i = 1, 2, \ldots, N$ **do**
  $Q^A_{i-1}(s, a) = \frac{1}{K} \sum_{k=1}^{K} Q(s, a; \theta_{i-k})$
  $y^i_{s,a} = \mathbb{E}_{\mathcal{B}}\left[r + \gamma \max_{a'} Q^A_{i-1}(s', a') | \, s, a\right]$
  $\theta_i \approx \text{argmin}_\theta \, \mathbb{E}_{\mathcal{B}}\left[(y^i_{s,a} - Q(s, a; \theta))^2\right]$
  Explore$(\cdot)$, update $\mathcal{B}$
end for
output $Q^A_N(s, a) = \frac{1}{K} \sum_{k=0}^{K-1} Q(s, a; \theta_{N-k})$

Figure 2. ADQN Algorithm. theta is weight, B is Experience Replay buffer, Explore(dot)is exploration procedure and y is value estimate at given state and action.

3. Approach

Since DQN and ADQN are wisely studied and compared by Anschel, Oron, Nir Baram, and Nahum Shimkin, we pay more attention on the effect of DDQN and ADDQN in this project.

3.1. DDQN

Double DQN is an algorithm that use two sets of deep neural network: one for action selection, and the other for action evaluation. DDQN prevents overoptimistic value estimates. On the opposite, the max operator in standard DQN uses same values to select and to evaluate an action, which makes it more likely to select overestimated values, resulting in overoptimistic value estimates. In a word, DDQN decouple the selection from the evaluation to prevent overestimation.

3.2. Averaged DDQN

The idea of double DQN is to use two sets of deep neural networks, one for action selection the other for action evaluation, and thus to prevent overoptimistic value estimates. And averaged DQN uses previous K learned Q values to generate current estimate of Q value to reduce the variance induced in training process. To combine these two methods together, we have ADDQN. ADDQN takes both advantages which are robust to variance and less prone to over optimistic estimation.

We can easily derive the value function of averaged double DQN by averaged DQN and double DQN given above.

Algorithm 3 DDQN

Initialize Q(s,a;θ), Q(s,a;θ') with random weights θ_0, θ'_0

for $i = 1, 2, \dots, N$ **do**

Pick arbitrarily from $\boldsymbol{\theta_{i-1}}$ **and** $\boldsymbol{\theta'_{i-1}}$

If pick $\boldsymbol{\theta_{i-1}}$

$$y^i_{s,a} = r + \gamma \operatorname*{argmax}_{a'} Q(s', a';\ \theta'_{i-1}) | s, a$$

$$\theta_i = \operatorname*{argmin}_{\theta} [(y^i_{s,a} - Q(s, a;\ \theta_{i-1}))^2]$$

Else

$$y^i_{s,a} = r + \gamma \operatorname*{argmax}_{a'} Q(s', a';\ \theta_{i-1}) | s, a$$

$$\theta'_i = \operatorname*{argmin}_{\theta} [(y^i_{s,a} - Q(s, a;\ \theta'_{i-1}))^2]$$

end for

Figure 3. DDQN Algorithm. theta and theta' are two sets of weights for selection and evaluation respectively and y is value estimate given state and action.

Algorithm 4 Averaged DDQN

Initialize Q(s,a;θ),Q(s,a;θ') with random weights θ_0, θ'_0

Initialize Experience Replay Buffer B

Initialize exploration procedure *Explore(·)*

for $i = 1, 2, \dots, N$ **do**

$$Q^{AD}_{i-1}(s, a) = \frac{1}{K} \sum_{k=1}^{K} Q(s, a; \theta_{i-k})$$

$$y^i_{s,a} = E_{\mathcal{B}}[r + \gamma \operatorname*{argmax}_{a'} Q^{AD}_{i-1}(s', a') | s, a]$$

$$\theta_i, \theta'_i = \operatorname*{argmin}_{\theta, \theta'} E_{\mathcal{B}}[(y^i_{s,a} - Q(s, a; \theta'))^2]$$

$$Explore(\cdot), \textbf{update } \mathcal{B}$$

end for

Figure 4. ADDQN Algorithmd.theta and theta' are two sets of weights for selection and evaluation respectively, B is Experience Replay buffer, Explore(dot)is exploration procedure and y is value estimate given state and action.

3.3. Theoretical Results

Compared with DQN, DDQN theoretically reduce overoptimism by decomposing the max operation into action selection and action evaluation, improving performance. Hence, the mean-max Q value of DDQN should be smaller than that of DQN while the average reward per episode is supposed to be larger than that of DQN. (Van Hasselt, 2016)

According to Anschel (2017), ADDQN will not only solve overestimation problem, but also reduce approximation error variance in the target values, compared with DQN. Hence, the mean-max Q value of ADDQN should also smaller than DQN. Besides, performance of ADDQN should be better than DQN, by which indicating the average score per episode of ADDQN is theoretically larger than that of DQN. Moreover, it should have a better convergence, going with a smoother and more stable line, compared with DQN and DDQN.

As for the computation cost, ADDQN goes first, following by DDQN and DQN. It suggests the rank of computation time.

4. Experiment Results

4.1. Mean Max Q (Overestimation)

Asterix shows that the Mean Max Q value of DQN is the largest, following by DDQN and then ADDQN. In fact, Q value of DQN is apparently larger than those of DDQN and ADDQN before the 600000th step while DDQN and ADDQN have relatively similar Q value. On the opposite, ADDQN converge the fastest, following by DDQN and then DQN. After DQN converge, Q in DQN are still slightly larger than Q in DDQN and ADDQN. DDQN converge right after ADDQN at the 200000th step. However, DQN goes to the peak at about the 280000th step and then finally, converge at approximately the 750000th step.

Besides, ADDQN also seem more stable with a smoother line compared with DQN. DDQN is also smooth but is not as smooth as ADDQN in 0.1 million steps figure.

The result proves that both ADDQN and DDQN cut down overoptimism. ADDQN have better reduction than DDQN. Moreover, it proves that ADDQN improve stability.

For Space Invaders, the Mean Max Q value of DQN is the largest, following by ADDQN and then DDQN. The difference with Asterix is that ADDQN is larger than DDQN in Space Invaders. DDQN converge the fastest while both ADDQN and then DQN do not converge in the 100000th step. The undesired result is mainly to be blamed to the insufficient training steps.

To be delighted, ADDQN is more stable with a smoother line compared with DQN and DDQN, which is consistent with the result of Asterix.

For Seaquest, ADDQN have the smallest Mean Max Q. Lines of DQN and DDQN is cross at about 40000th step. The average Q is 4.00 in DQN, 3.92 in DDQN and 0.68 in

ADDQN. We could also say that the Mean Max Q value of DQN is the largest, following by DDQN and then ADDQN in Seaquest, like that in Asterix. The crossing phenomenon also results from the insufficient training steps. The 100000 steps we train may be the initial part of a sufficient training process.

Clearly telling from Figure 8, ADDQN is more stable with a smoother line compared with DQN and DDQN, which is consistent with results of Asterix and Space Invaders.

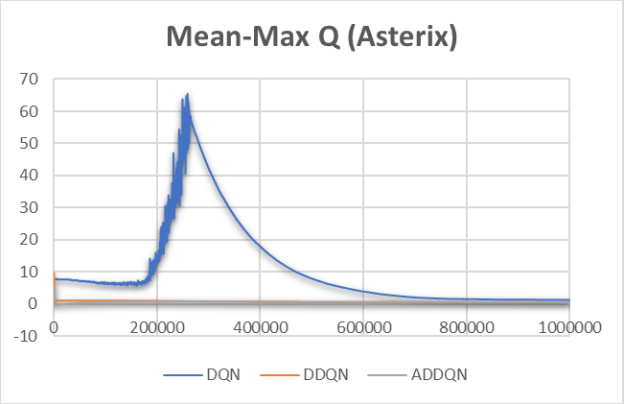

Figure 5. Overestimation can be learned by Mean Max Q values. Game: Asterix. Algorithms: DQN, DDQN and ADDQN. Steps:1 million.

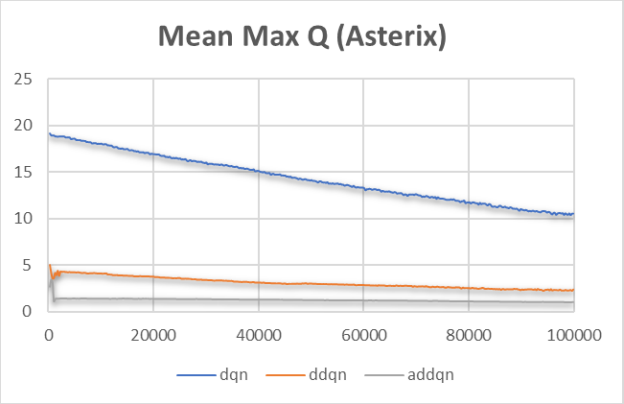

Figure 6. Overestimation can be learned by Mean Max Q values. Game: Asterix. Algorithms: DQN, DDQN and ADDQN. Steps:0.1 million

4.2. Average Reward (Performance)

The average reward of DQN, DDQN and ADDQN in Asterix are 204.86, 217.50 and 219.03, respectively. The average reward of DQN, DDQN and ADDQN in Space Invaders are 203.99, 210.42 and 214.67, respectively. The average reward of DQN, DDQN and ADDQN in Seaquest are 168.09, 141.90 and 159.88, respectively. For Asterix and Space Invaders, both ADDQN have the highest average reward, following by DDQN and then DQN. Those results tell that DDQN and ADDQN ameliorate performance and ADDQN perform better than DDQN. In Seaquest, ADDQN also perform better than DDQN with larger average reward, but both are smaller than the average reward of DQN.

None of DQN, DDQN and ADDQN coverages in three games, which may result from insufficient numbers of episodes as discussed before.

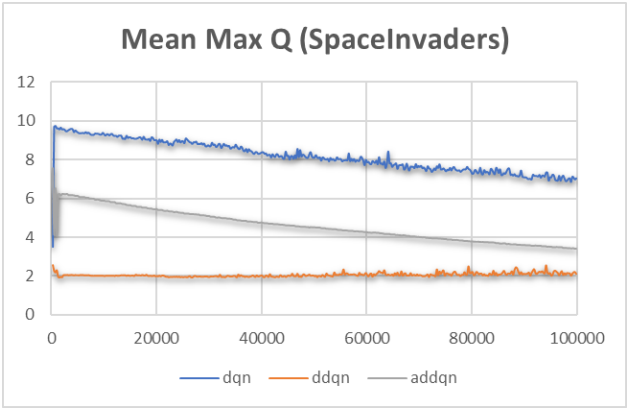

Figure 7. Overestimation can be learned by Mean Max Q values. Game:Space Invaders. Algorithms: DQN, DDQN and ADDQN. Steps:0.1 million

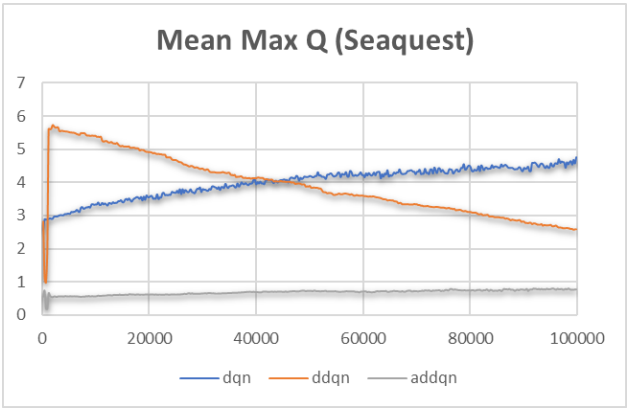

Figure 8. Overestimation can be learned by Mean Max Q values. Game:Seaquest. Algorithms: DQN, DDQN and ADDQN. Steps:0.1 million

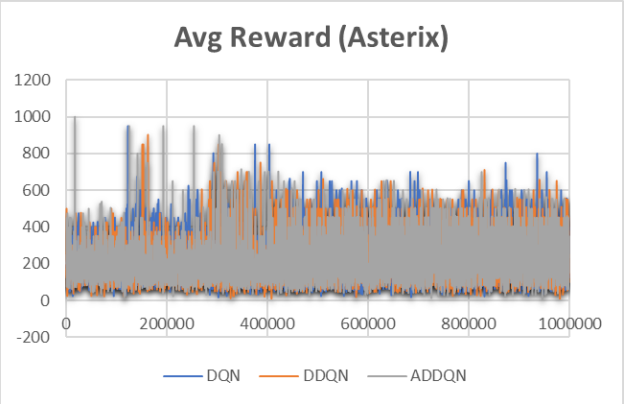

Figure 9. Performance can be learned by Average Rewards. Game: Asterix. Algorithms: DQN, DDQN and ADDQN. Steps:1 million.

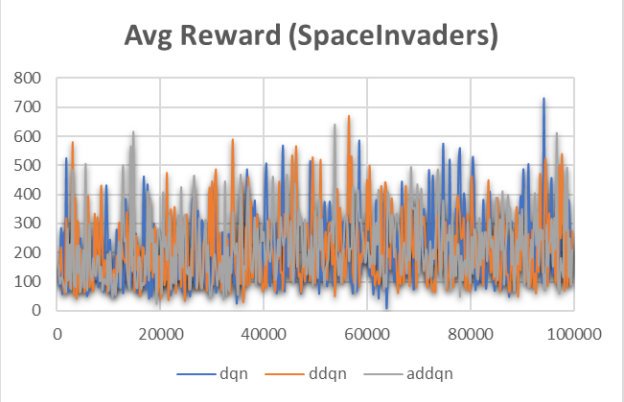

Figure 10. Performance can be learned by Average Rewards. Game: Space Invaders. Algorithms: DQN, DDQN and ADDQN. Steps:0.1 million.

4.3. Time (Computational Cost) for 100 thousand Steps

Running time/s	Asterix	Space Invaders	Seaquest
DQN	2950	3146	2966
DDQN	3269	3445	3413
ADDQN	3476	3564	3445

In 100 thousand steps for three games, all ADDQN are the most time-consuming, following by DDQN and DQN. However, the processing time of ADDQN and DDQN are closer with the average rate 0.97 while the average rate between ADDQN and DQN is 0.86.

Hence, we can not consider the computation cost if ADDQN is much better than DDQN on other aspects.

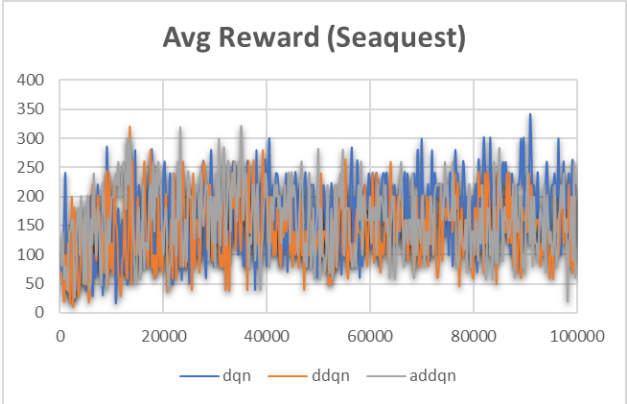

Figure 11. Performance can be learned by Average Rewards. Game: Seaquest. Algorithms: DQN, DDQN and ADDQN. Steps:0.1 million.

5. Conclusion

We implemented DQN, Double-DQN and Averaged Double-DQN algorithms on three Atari games: Asterix, Space Invaders and Seaquest. Now we present the results of Mean Max Q, Avg Reward and Time for three games. We have compared those three different algorithms with respect to performance, help in overestimation and computational cost, as shown in the experimental result section.

In conclusion, both ADDQN and DDQN solve the overestimation problem. ADDQN usually have better reduction than DDQN. At the same time, ADDQN apparently improve stability of training. Moreover, both DDQN and ADDQN ameliorate performance, and ADDQN usually perform better than DDQN. Besides, ADDQN have a slightly larger computational cost than DDQN, but we could ignore the small computation-cost difference if ADDQN is much better than DDQN on other aspects.

References

Anschel, Oron, Nir Baram, and Nahum Shimkin. ”Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning.” In International Conference on Machine Learning, pp. 176-185. 2017.

Bellemare, Marc G., Yavar Naddaf, Joel Veness, and Michael Bowling. ”The Arcade Learning Environment: An evaluation platform for general agents.” J. Artif. Intell. Res.(JAIR) 47 (2013): 253-279.

Li, Chong. ”Function Approximation.” ”Deep Reinforcement Learning.” Class Note. 2017.

Sutton, Richard S and Barto, Andrew G. Reinforcement Learning: An Introduction. MIT Press Cambridge, 1998.

Van Hasselt, Hado V. "Double Q-learning." In Advances in Neural Information Processing Systems, pp. 2613-2621. 2010.

Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-Learning." In AAAI, pp. 2094-2100. 2016.

Contributions

	Jianhuan	Xiaoning	Yuwei
Theory	5	4	5
Coding of DDQN	3	5	3
Coding of ADDQN	2	5	5
Result and Discussion	5	3	4
Report	5	4	4

The first row presents three team members; the first column shows tasks of this project; then the table shows rates of contribution of each team member with respect to different tasks; each rate is out of 5.