Vision-based Deep Reinforcement Learning: Playing Atari Games

105-1 Artificial Intelligence Final Project

學生：林昱安 (B01901066)、李佳軒 (B01901073)　　指導教授：于天立

Abstract

Recent researches have made many success of utilizing deep representation to extend reinforcement learning. These algorithms have allowed agent to play complex games such as Atari video games and GO, which is able to achieve human level performance. Still, different design of these algorithms lead to different behavior of their agent while playing games. Our research investigated three reinforcement learning algorithms based on deep learning, and compared their behavior and performance.

I. Introduction

Making computer play games has long been an interesting problem. The development of artificial intelligence has provided various ways to solve the problem, including searching algorithms and learning algorithms, which were proved to be successful in simpler games both theoretically and practically. However, making intelligent agent to achieve human-level control in complex games such as GO or video games has still remained as a challenging task.

In recent years, the research of deep learning and reinforcement learning has provided algorithms which make major success of making intelligent agent achieve beyond human-level control in complex games. By transforming games into a sequential decision problem, it can be modeled by Markov Decision Process (MDP), where the games are represented as a sequence of states. Q-learning algorithm (Watkins, 1989) is one of the most popular reinforcement learning algorithm which solve the sequential decision problem. However, applying Q-learning algorithm to complex games often face difficulties such as extracting features from high dimensional input, dealing with large amount of states and partial observable environment, which limited its performance. The deep reinforcement learning (DQN) algorithm (Mnih et al., 2015) combines Q-learning with deep neural network and successfully enhance their performance. Moreover, these algorithms provided end-to-end solutions which are flexible enough to solve various tasks other than one single game. The DQN agent was tested on the Atari 2600 games, which surpass the performance of both previous research and professional human level.

Based on the recent success of DQN algorithm, more researches have further improved its stability and performance. The double DQN (DDQN) learning algorithm of van Hasselt at el. (2015) have extend the DQN to solve the overestimation problem in Q-learning. The dueling network architecture of DQN (Wang et al., 2016) separate the estimation on state value function and action advantage function (dependent on state), and has shown improvement in performance on the Atari 2600 games.

In the experiment, we select three Atari games and evaluate their performance from agent with DQN, DDQN and Dueling DQN algorithms respectively. We also perform experiment on parameters to understand how DQN agent's behavior dependence on them. Result from these experiments illustrate the different characteristics between these algorithms, and may pave the way to future research.

II. Background

II.I. Reinforcement Learning

Consider a sequential decision making setup, in which an agent interact with an environment E. The agent perceives a sequence of state at time t, $s_t \in S$, and then chooses an action $a_i \in A$ among all possible action A to transfer to the next state s_{t+1}. At each state s, the agent will receive a reward $R(s)$ from the environment. The sequential decision problem is to decide policy $\pi(s)$, which is the best action for the agent to behalf at state s, to receive maximum reward when the state sequence terminated.

Q-learning algorithm is a popular algorithm in the field of reinforcement learning, which aims to solve the sequential decision problem. For an agent behaving according to a given policy π, the value of the state-action pair (s, a) are defined as follow:

$$Q_\pi(s,a) \equiv E[R_1 + \gamma R_2 + \cdots | S_0 = s, A_0 = a, \pi] \quad (1)$$

where γ is a discount factor that trades off the importance of immediate and later rewards. To determine the optimal policy, we easily choose the action with maximum Q-Value given a state.

For a partial observable environment, we are unable to know the transition probability between states, and we may apply the Temporal difference (TD learning) approach. Each time the agent visit the state s, choose an action a and receive a reward $R(s)$, the TD learning approach updates the Q-Value with the following formula:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s', a') - Q(s,a)) \quad (2)$$

Most interesting problems ate too large to learn all action values in all state separately, and are too costly to store the entire Q-value table. Instead, we can learn a parameterized value function $Q(s, a; \theta_t)$. Instead of updating the Q-value, we update the parameter θ at step t with the following formula:

$$\theta_i \leftarrow \theta_i + \alpha(R(s) + \gamma \max_{a' \in A(s')} \hat{Q}_\theta(s', a') - \hat{Q}_\theta(s,a)) \frac{\partial \hat{Q}_\theta(s,a)}{\partial \theta_i} \quad (3)$$

With enough step, the parameter θ will gradually converge. We are then able to compute Q-value and optimal policy accordingly.

II.II. Deep Reinforcement Learning

Traditional Q-learning are facing several challenges in playing complex games, such as video games like Atari 2600. When we use high dimensional data as input, such as images, we need automatic feature extractions to approximate the value function $Q(s, a; \theta_t)$. However, nonlinear function approximater to estimate $Q_\theta(s, a)$ in known to be unstable and easy to diverge. The instability is caused by the high-correlation characteristic of sequential data, which make it difficult to learn the parameters θ.

The deep reinforcement learning algorithm has aimed to solve the problem. A DQN is a multi-layered neural network that for a given state s, which is a represented by images, outputs a vector of action values $Q(s, a; \theta_t)$. Where θ is the parameters of the network. Two important ingredients of DQN algorithms are target Q-network and experience reply.

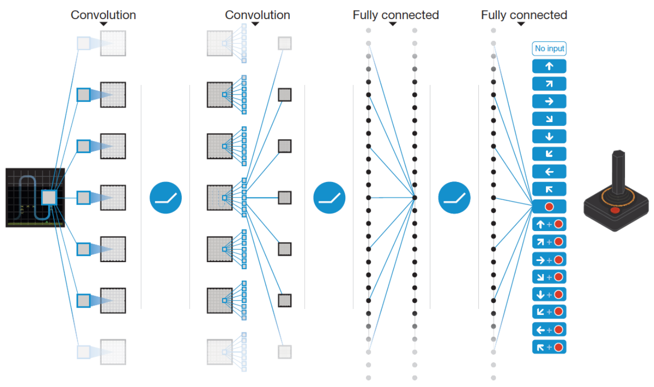

Figure 1: Illustration of convolutional neural network as value function appriximater. With image as input, the network performs supervised learning on Q-Value with nonlinear estimations with parameters stored as weights in the network.

For target network, the parameters of the network θ^- are being used to compute Q-value, for all steps before updated to θ. For memory replay, observed transitions (s_t, a_t, r_t, s_{t+1}) are stored in memory, and sampled uniformly for the use of training after the agent has make several steps. The learning task of the network is to minimize the loss:

$$L_i(\theta_i) = E_{(s,a,r,s')\sim U(D)}[\left(r + \gamma \max_{a'} Q\left(s', a'; \theta_i^-\right) - Q(s, a; \theta_i)\right)^2] \tag{4}$$

The DQN algorithm has dramatically improve the performance on Atari games.

II.III. Double Deep Q-networks

The max operator in Q-learning is known to overestimate the Q-value. Since the estimated Q-value may differ from the true Q-value, computing target Q-value in this manner may cause overoptimism due to estimation errors. This problem is especially severe in DQN since the estimation errors may be high in the initial process. The improved Double DQN (DDQN) algorithm decouple the selection from the evaluation to mitigate the problem. DDQN use the following target:

$$y_i^{DDQN} = r + \gamma Q(s', arg \max_{a'} Q\left(s', a'; \theta_i\right); \theta^-) \tag{5}$$

DDQN are same as DQN, but use target y_i^{DDQN} to evaluate the loss function, which reduce the overestimate problem.

II.IV. Dueling Network Architectures for Deep Reinforcement Learning

The dueling network architectures for DQN represents two separate estimations: one for the state value function and one for the state-dependent action advantage function. The key insight is that, the choice of actions may be unimportant in some states, but very important in other states. By separating the estimation with different network, we can generalize learning across actions. Another advantage is that the parameters of state value function are updated when the state is visited, while in original Q-learning each Q-value is updated only when specific state and action are given. This advantage makes Dueling DQN more efficient then original DQN.

The dueling DQN network structure are similar to DQN network structure, which use convolution neural network to extract feature from images. While in dueling structure, we evaluate state value function and advantage function of action separately and combine them to acquire Q-value, $Q(s,a) = V(s) + A(s,a)$.

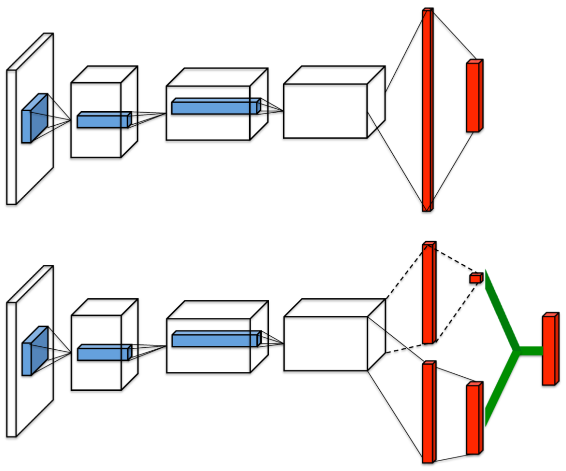

Figure 2: Comparison of simple DQN network and dueling DQN network. The simple DQN network (above) has only one stream, while dueling DQN network has two stream to evaluate state value function and action advantage function.

III. Experiments and Results

To understand how DQN algorithm has successfully play video games, and how double DQN and dueling DQN has improved it performance, we perform experiments on three of the Atari games: Seaquest, MsPacman and Tennis.

III.I. Comparison of DQN based algorithms with game Seaquest

We trained the agent with DQN, DDQN and Dueling DQN algorithms with video game Seaquest. The training took around 6 Million of steps, and was evaluated by average reward per episode.

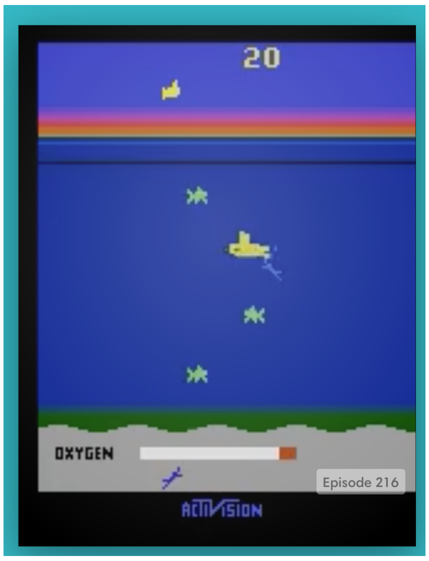

Figure 3: The Atari game Seaquest. The sublime will earn rewards when shot a fish. Each time it touches one fish or run out of oxygen will lose one life, while the game terminates when it loses all three lifes.

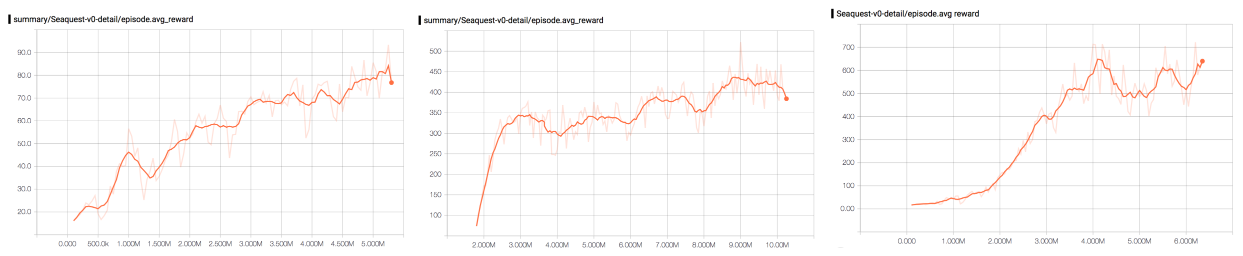

Figure 4: Average reward per episode while training. The dueling DQN (right) outperforms double DQN (middle) and simple DQN (left) in training stage.

Figure 4 shows that dueling DQN has the best average reward comparing to the other two methods within 5 million of steps, whose average reward are two times greater than double DQN and four times greater than simple DQN. In all three algorithms, there exist oscillation behavior in training progress, and struggles to achieve higher reward after a number of steps.

III.II. Comparison between testing performance of DQN based algorithms

UNIT : Episode_Reward				
Reward Result	**Average(on100Episodes)**	**Max**	**min**	**Model_STEP_USED**
Sequest_DQN	**150**	**340**	**60**	**5.25M**
Sequest_Double	**1943**	**4220**	**400**	**9M**
Sequest_Dueling	**2230**	**8520**	**0**	**7.85M**
Sequest_Dueling_greedy	**108**	**420**	**0**	**9.4M**
Pacman_DQN	**881**	**1590**	**360**	**2.2M**
Pacman_Double	**1406**	**2790**	**430**	**1.8M**
Pacman_Dueling	**1101**	**3150**	**190**	**1.25M**

Table 1: Average reward per episode in testing. The dueling DQN outperforms double DQN and simple DQN in testing stage.

The experiment result is quite interesting. There is no absolute winner among the three models. The only thing for sure is that Dueling and Double are both better than DQN. This fits our assumption. The original DQN only performs well on frequent rewarding and hard-to-die games. But in Sequest and Pacman, there are a lot of enemies that can cause the agent to die. The game is much harder and unpredictable.

Agent trained by dueling gets the max episode reward in both games and also the min ones. Agent trained by double is the most stable one. Reward range is the smallest. This is not surprising because the model uses two networks to make a more reasonable action to avoid extreme behavior.

In Sequest, the ocean is often “empty”. And thus, it is a useless state. No matter how the agent does it’s meaningless. (Doesn’t help to gain reward) Also in this game the agent needs to flow up to the surface before the oxygen runs out. So, the agent needs to keep track of how many steps have passed since it sinks. Making the state evaluation more critical. Dueling is especially good at evaluating state than Double and normal DQN.

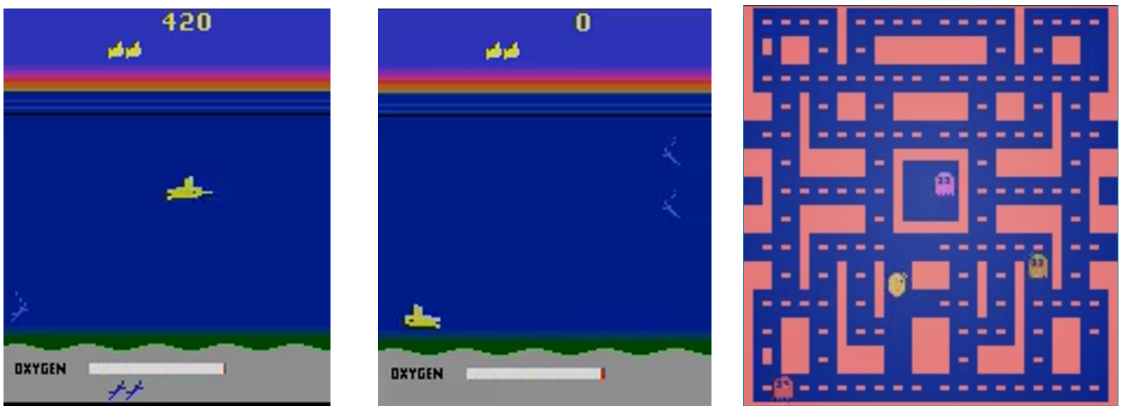

Figure 5: Screen shoots of games. The Seaquest game (left, middle) and Pacman game (right)

One interesting thing is that DQN rushes to the left bottom of the ocean every time it sinks. Then remains there until it needs oxygen. I think this is a local optimum. The agent learns to play it safe. In this way, it only has to deal with the enemy from left and right. It gives up the rewards it can get if acts more aggressive.
This game of Pacman is more difficult to learn. The map is complicated and there are four aggressive agents that will approach the agent actively. And there are secret tunnels. Overall, three models don't perform well. Maybe they require more training steps. The agent trained by dueling is again the most aggressive one, while Double DQN agent tends to play it safe.

For videos of our testing result, please visit the following link:
https://www.youtube.com/channel/UCNntBC7gBhzbHRxOBN6brjA

IV. Conclusion and Future Work

Our research experimented the three reinforcement learning algorithms extended by deep learning, including DQN, double DQN and dueling DQN algorithms. Result of our experiment on the game Seaquest and Pacman shows that Dueling DQN outperforms the other two methods, but double DQN was the most stable algorithms among the three.

V. Reference

1. Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *Nature* 518.7540 (2015): 529-533.
2. Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep reinforcement learning with double Q-learning." *CoRR, abs/1509.06461* (2015).
3. Wang, Ziyu, Nando de Freitas, and Marc Lanctot. "Dueling network architectures for deep reinforcement learning." *arXiv preprint arXiv:1511.06581*(2015).