Deep Reinforcement Learning on Atari 2600

T Manoj, J Rohit, A Sai Sashank & Asst. Prof. BJ Praveena
Dept. of Computer Science and Engineering, Matrusri Engineering College Saidabad, Telangana, India.

Abstract:- In reinforcement learning, the traditional Q Learning method solves the game by iterating over the full set of states. Using the Q-Table to implement Q-Learning is fine in small discrete environments. However often we realize that we have too many states to track. An example is Atari games, that can have a large variety of different screens, and in this case, the problem cannot be solved with a Q-table. This paper uses a deep neural network instead of a Q-table to solve it. Atari games are displayed at a resolution of 210 by 160 pixels, with 128 possible colors for each pixel. This is still technically a discrete state space but very large to process. To reduce this complexity, we performed some minimal image preprocessing. Finally, from the experimental results, it is concluded that DQN can make the agent get high scores from Atari game, and experience replay can make the model training better.

Keywords:- *Deep Reinforcement Learning; Artificial Intelligence; Image Processing;*

I. INTRODUCTION

The combination of Reinforcement Learning and Deep Learning is Deep Reinforcement Learning (DRL). Since the improvement of computer calculation power and processing technology, deep learning has gained significant advantage than traditional methodsin the field of artificial intelligence. To use reinforcement learning successfully in situations approaching real-world complexity, however, agents are confronted with a difficult task, they must derive efficient representations of the environment from high- dimensional sensory inputs, and use these to generalize past experience to new situations. In reinforcement learning the agent does not have knowledge on what actions to take instead, the agent learns from the consequence of its actions. Deep learning has strong perceptual ability but it is weak in decision making. In contrast, reinforcement learning performs well in decision-making, but has weak perceptual ability [1]. Therefore, combination of these two provide a way to solve the problem of perceptual decision-making in complex systems.

Here we use recent advances in training deep neural networks to develop a novel artificial agent, termed a deep Q-network, that can learn successful policies directly from high-dimensional sensory inputs using end-to-end reinforcement learning on Atari 2600 games [10].

II. PRELIMINARIES

In Q learning, the Q function, also called a state-action value function, specifies how good an action a is in the state s. Most of the environments have very large number of states and, in each state, we have a lot of actions to try. It would be time-consuming to go through all the actions in each state. A better approach would be to approximate the Q function with some parameter θ. We can use a neural network with weights to approximate the Q value for all possible actions in each state. As we are using neural networks to approximate the Q function, we can call it a Q network.

Assume the environment is in a state s from the state space S. The agent takes action a from the action space A by obeying the policy π (a | s). A may be discrete or continuous. When an action is performed, the agent transitions into a new state s' receiving a scalar reward r. If for every action, the reward and the next state can be observed, we can formulate the following iterative algorithm to learn the Q value:

$Q(s,a)$ will be updated follow:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r - Q(s,a)) \tag{1}$$

And if the round ends, it will follow:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\,[r + \gamma \max_{a'\in A} Q(s',a') - Q(s,a)] \tag{2}$$

Eq. (2), which is the maximum sum of rewards r discountedby γ ($0 < \gamma \le 1$) at each time step, achievable by a policy π (a | s), after making an observation s and taking an action a with learning rate α ($0 < \alpha \le 1$).

The optimal action-value function obeys the Bellman equation, following the intuition: if the optimal value $Q(s', a')$ of the sequence s' at the next time-step was known for all possible actions a', then the optimal strategy is to select the action a' maximizing the expected value of $r + \gamma Q^*(s', a')$ [10]. In the Q-Learning, a policy called ε - greedy, that follows a greedy strategy with probability 1 - ε and selects a random action with probability ε. ε will continue to decrease with continuous training.

The algorithm of DQN is presented as Figure 1.

```
Algorithm 1 Deep Q-Network
Initialize replay memory M to capacity N
Initialize the Q function with random weights $\theta$
for step = 1 to $step_{total}$ do
    Take the current state $s$ from the environment
    Select an action $a$ using policy $\pi(a \mid s)$
    if random probability $\le \varepsilon$ then
        Set $a = rand_{a \in A}$
    else
        Set $a = \max_{a \in A} Q(s', a'; \theta)$
    end if
    Execute action $a$ and observe the reward $r$ and new state $s'$
    Store transition $(s, a, r, s')$ in M
    Sample random mini batch of transitions $(s, a, r, s')$ from M
    if state is terminal state then
        Set $y = r$
    else
        Set $y = r + \gamma \max_{a \in A} Q(s', a'; \theta)$
    end if
    Perform a gradient descent step on loss
end for
```

Fig. 1. The Algorithm of DQN.

III. PROPOSED METHOD

The resolution of the Atari game is 210 x 160 as shown in Fig. 2, it will clearly take a lot of computation and memory if we feed the raw pixels directly. So, we down sample the pixels to 84 x 84 and convert the RGB values to grayscale values and we feed this pre-processed game screen as the input to the model.

Pong is a table tennis-themed arcade video game featuring simple two-dimensional graphics, manufactured by Atari and originally released in 1972. In Pong, one player scores if the ball passes by the other player. An episode is over when one of the players reaches 21 points. In the OpenAI Gym framework version of Pong, the Agent is displayed on the right and the enemy on the left. let’s think carefully if with this fixed image we can determine the dynamics of the game. There is certainly ambiguity in the observation, we cannot know in which direction the ball is going.

The solution is maintaining several observations from the past and using them as a state. In the case of Atari games, the authors of the paper suggested to stack 4 subsequent frames together and use them as the observation at every state. For this reason, the preprocessing stacks four frames together resulting in a final state space size of 84 x 84 x 4. The final input to the neural network is shown in Fig. 5, which are 4 stacked grayscale images of 84 x 84 pixels each. After the images are processed, the neural network needs to be built.

The first hidden layer convolves 32 filters of 8x8 with stride 4 with the input image and applies a rectifier nonlinearity. The second hidden layer convolves 64 filters of 4x4 with stride 2, again followed by a rectifier nonlinearity. This is followed by a third convolutional layer that convolves 64 filters of 3x3 with stride 1 followed by a rectifier. The final hidden layer is fully-connected and rectifier units. The output layer is a fully- connected linear layer with a single output for each valid action [10]. The number of valid actions in pong are 6.

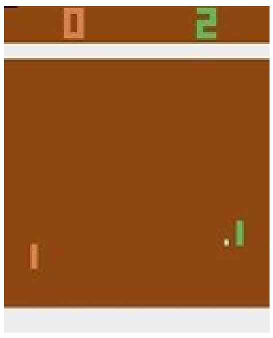

Fig. 2. The RGB image with 210 x 160 pixels

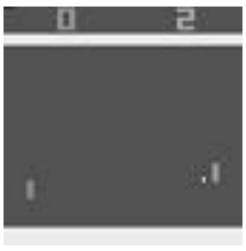

Fig. 3. The GRAY image with 84 x 84 pixels

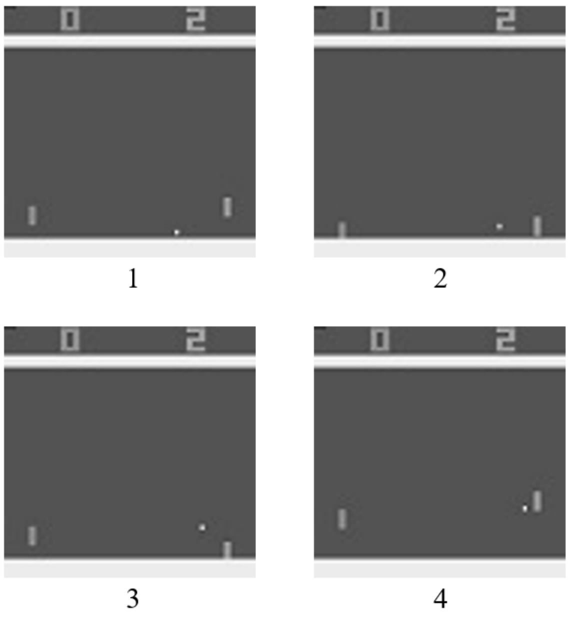

Fig. 4. The four subsequent frames of pong with 84 x 84 pixels each.

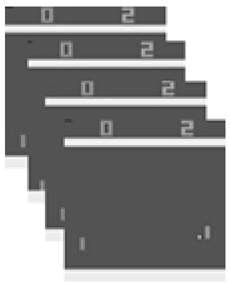
Fig. 5. Final input size (84 x 84 x 4).

IV. EXPERIMENTAL RESULT

In this paper, we demonstrate that DQN architecture can successfully learn control policies in the pong environment with only very minimal prior knowledge, receiving only the pixels and the game score as inputs, the reward structure of the pong is as follows, Reward is 0 for every frame and +1 for every ball missed by other player. The reward is -1 when ball is missed by agent. Then, In the process of training the model, it is possible that there will be many cases of ineffective training.

Following previous approaches to playing Atari 2600 games, we also use a simple frame-skipping technique. More precisely, the agent sees and selects actions on every 4th frame instead of every frame, and its last action is repeated on skipped frames.

The values of all the hyperparameters and optimization parameters were selected by performing an informalsearch on the game Pong. The following are values and descriptions of hyperparameters used in training, the discount factor $\gamma = 0.99$, the planned training is 1,000,000 steps, the initial exploration (ε initial value) and the final exploration (ε final value) are 1.0 and 1e-2, respectively, and the random sampling batch size is 32.

Note during the experiment, the data of the average scores and the weights of neural network have been stored.

Then the agent model was put to test on pong game, the highest scores were recorded. The Fig. 6 shows the data of average rewards over 1,000,000 steps trained by the agent.

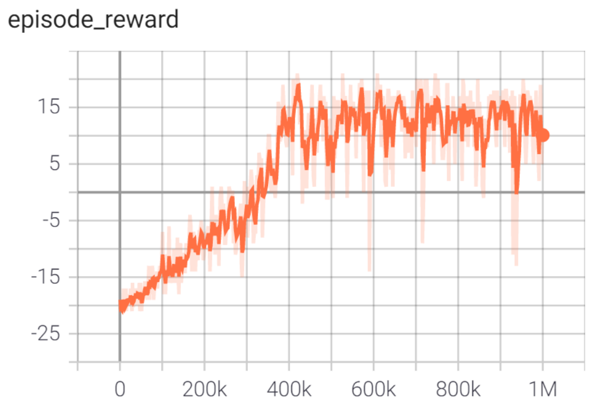

Fig. 6. The graph plot of episode reward over 1M steps.

V. CONCLUSION

In this paper, the agent model has been created to verify the effectiveness of experience replay. From the training result (the agent's average score-per-step, see Fig. 6), it can be seen that, the effect of training can grow very slowly after about 300,000 steps. Then the agent learns faster and more steadily.

We used the network architecture, hyperparameter values and learning procedure taking high-dimensional data as input (visual images), and the number of actions available in each game with only very minimal prior knowledge. Our method was able to train large neural networks using a reinforcement learning and stochastic gradient descent in a stable manner.

Finally, it can be concluded that the DQN algorithm combined with deep convolution neural network and some image processing methods can make the agent play the video game well and the experience replay can improve the training quality.

REFERENCS

[1]. R. Tan, J. Zhou, H. Du, et al., An modeling processing method for video games based on deep reinforcement learning, in: 2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference, ITAIC, IEEE, 2019, pp. 939–942.

[2]. SILVER D, HUANG A and et al. Mastering the game of Go with deep neural networks and tree search. Nature, 2016.

[3]. Wu Xiru, Huang Guoming and Sun Lining. Fast visual identification and location algorithm for industrial sorting robots based on deep learning. Jiqiren/Robot, November 1, 2016.

[4]. Riedmiller, M. Neural fitted Q iteration - first experiences with a data efficient neural reinforcement learningmethod. Mach. Learn.: ECML, 3720, 317–328 (Springer, 2005). & Hutter, M. Universal Intelligence: a definition of machine intelligence. Minds Mach. 17, 391–444 (2007).

[5]. Moore, A. & Atkeson, C. Prioritized sweeping: reinforcement learning with less data and less real time. Mach. Learn. 13, 103–130 (1993).

[6]. O'Neill, J., Pleydell-Bouverie, B., Dupret, D. & Csicsvari, J. (2010) Play it again: reactivation of waking experience and memory. Trends Neurosci., 33, 220– 229.

[7]. Lange, S. & Riedmiller, M. Deep auto-encoder neural networks in reinforcement learning. Proc. Int. Jt. Conf. Neural. Netw. 1–8 (2010).

[8]. Richard S. Sutton and Andrew G. Barto. Reinforcement Learning. MIT Press, 1 edition, 1998.

[9]. Eric Wiewiora. Potential-based shaping and q-value initialization are equivalent. J. Artif. Intell. Res. (JAIR), 2003.

[10]. MNIHV, KAVUKCUOGLUK, SILVERD, et al. Human-level control through deep reinforcement learning[J]. Nature,2015.