Deep Q-Learning AI for Single-Player Atari Games

Xiaoming Wu, Yangfan Yu
wu.xiaom@husky.neu.edu, yu.ya@husky.neu.edu

Abstract

The main goal of the project is to implement an agent to play s single-player video games mainly by using deep reinforcement learning technique. What we are trying to improve is the generality of the agent: to play different kinds of single-player video games.

1 Introduction

1.1 Motivation

As the development of deep learning techniques, especially for Convolutional Neural Network's (a.k.a. CNN) achievements in computer vision field, it is productive for reinforcement learning to play video games by using deep-learning.

The experience in playing video games also has great meaning in reality, for example, the actions of those single player games like Breakout, Cartpole are very similar to that of industrial robots, both work in a simple situation, the goal is also to improve the performance of their actions. Solving more complicated environments like driverless cars' environment can also benefit from it.

Extracting important pixels, estimating rewards from observations of random behavior without prior experience and applying the algorithm to different games who have different state, actions, rewards are the main challenges.

1.2 Approach

The algorithm is Deep Q-Learning (a.k.a. DQN) with experience replay (based on Deep Mind paper on 2015), for each time of training, the training data is saved and can be loaded for next training. Downsampling is adopted for simplifying the process of pixel data. CNN is used to deal with the image feature part of the video games and also a key part of our neural network.

2 Related Work

2.1 Algorithms

Many reinforcement learning(RL) algorithms have been applied to game playing, such as SARSA (Rummery and Niranjan, 1994), Q-learning(Watkins, 1989), and Actor-Critic(Barto et al., 1983).

The Markov Decision Process(MDP)(Bellman, 1957) is a formation for reinforcement learning, which defines five aspects to formulate a RL problem: S (the agent's state space), A (the agent's action space), T(s, a ,s') (the transition dynamics from s with action a), R(s, a, s') (reward function), ε (the set of terminal states).

Q-function is the key concept for MDP, the Q-function for the optimal policy can be expressed as below:
$Q^*(\mathrm{s},\mathrm{a}) = \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma max_{a'} Q^*(s',a)]$ [1].

In Q-Learning, an agent begins with an arbitrary estimate (Q_0) of Q* and iteratively improves its estimate by taking arbitrary actions in the environment, observing the reward and next state, and updating its Q value estimate according to[2]:

$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha_t[r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a') - Q_t(s_t, a_t)]$[3].

Human-level control through deep reinforcement learning (Mnih et al., DeepMind, 2015) used deep neural networks to develop a deep Q-network(DQN), which combined Q-learning with deep neural networks, specifically, deep convolutional neural network was used to approximate the optimal action-value function(Q-value). To address the instability of reinforcement learning, DeepMind used experience replay in which they stored the agent's experiences at each time step, in a dataset D over many episodes, and in learning process, they extracted samples of experiences randomly from pool of samples.

2.2 Framework and Benchmark

Several benchmarks have been released for researchers to compare algorithms, for example, ALE (Bellemare et al.

2013) for Atari 2600 games, RL Lab for continuous control, and OpenAi Gym. OpenAi Gym has provided environments over many different types of games and tasks, we are using the Atari environments provided by Gym for our project.

GENERAL VIDEO GAME AI: COMPETITION, Challenges, and Opportunities (DiegoPerez-Liebana, et al., 2016) proposed a General Video Game AI Framework, in which a Video Game Description Language is employed to define games in a general way.

Our project is mainly based on the DeepMind research' on Atari games, and we run our AI on Google Compute Engine.

3 Approach

3.1 Deep Q-Learning

Deep Q-Learning(DQN) (DeepMind, 2015) used deep convolutional neural networks for Q-function approximation and introduced experience replay[4] in which they stored experiences at each time step and applied Q-learning updated from the samples of experiences selected randomly from the pool of stored samples during the inner loop of the algorithm.

Pseudocode for DQN copied from DeepMind's research is shown below in algorithm 1:

Algorithm 1 Deep Q-learning with experience replay:

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function ˆ Q with weights $\theta^{-} = \theta$
for episode 1, M **do** Initialize sequence s1 = $\{x_1\}$ and preprocessed sequence $\varphi 1 = \varphi(s_1)$
for t = 1, T **do**
With probability ε select a random action a_t
otherwise select at = argmaxa Q(φ(st), a; θ)
Execute action at in the emulator and observe reward r_t and image x_{t+1}
Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\varphi_{t+1} = \varphi(s_{t+1})$
Store experience $(\varphi_t, a_t, r_t, \varphi_{t+1})$ in D
Sample random minibatch of experiences $(\varphi_j, a_j, r_j, \varphi_{j+1})$ from D
Set $y_j = r_j$ if episode terminates at step j + 1
Set $r_j + \gamma \max_{a'} \hat{Q}(\varphi_{j+1}, a';\theta^{-})$ otherwise
Perform a gradient descent step on $(y_j - Q(\varphi_j, a_j; \theta))^2$
with respect to the weights θ
Every C steps reset ˆQ = Q
end for
end for

3.2 Preprocessing

Atari games are presented as 210·160 images with RGB color, low-resolution but still huge input to process and compute. Therefore, it is necessary to reduce the image size and color complexity.

During preprocessing, we resize the input image form 210·160·3 to 84·84. So the input the number of variables are reduced from 100,800 to 7056. And in those Atari games, color is not a key parameter, what matters are the relative position of objects and the number of objects, a gray-scale image can still well render the games. To simplify problem, we convert the images from RGB (128) to gray-scale (4).

3.3 Exploration: ε - greedy

It is important to avoid the agent keep moving back and forth in a small action space, hence it is necessary to have a good exploration strategy to try new actions.

We use ε greedy to do exploration, the agent chooses random actions with probability ε, and takes actions according to current polity (greedy exploitation) with probability 1- ε. The ε needs to be reduced over time since the learned policy is better over time. We start decaying ε from 1 and end by 0.1 over time, ε decays (1-0.1)/number of steps for each step.

3.4 Experience Reply

The DQN stores experiences at each time step and uses prior experience for new training procedure. However, consecutive sequence of observations are strongly correlated, data distribution can be greatly changed due to a small update to Q. This will let the agent take oscillating actions and be stuck in poor local minimum[5].

We only store past 4000 experience tuples in the replay memory. In order to avoid potential overfitting by taking past experience, we take uniformly random samples from experience dataset D to update Q value, in this way, correlations can be reduced. This random policy is run for 50000 steps before any gradient descent to avoid overfitting.

3.5 Deep Neural Networks

Our neural network consists of three convolutional layers and two fully connected layers. The architecture is show as the figure 1 below.

The input to the network is an 84·84·4 image preprocessed, the first hidden convolutional layer convolves 32 filters of 8·8 with stride 4 with the input image and applies a rectifier nonlinearity[6]. The second hidden layer convolves 64 filters of 4·4 with stride 2, again followed by a rectifier nonlinearity. A third convolutional layer that convolves 64 filters of 3·3 with stride 1 followed by a rectifier. The final hidden layer is a fully-connected layer which consists of 512 rectifier units. The output layer is a fully-connected linear layer with a single output for each valid action of the trained game[7]. Each fully connected layer contains 512 neurons.

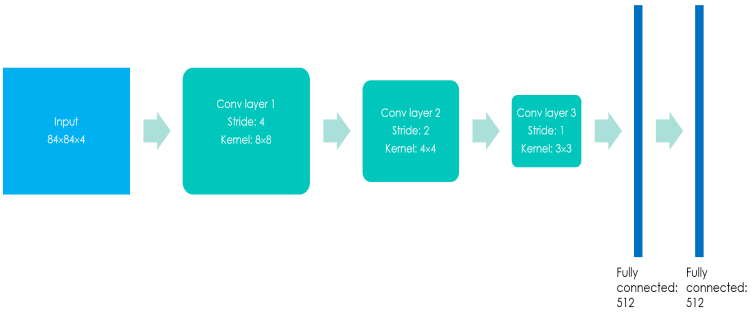

Figure 1: Neural Network Architecture

4 Evaluation

We test our algorithm using three different Atari games:

Game	Action Space
Freeway-v0	3
Breakout-v0	4
Pong-v0	6

Table 1: Action Space of Game

We compare human player, random policy agent with our DQN agent.

4.1 Human performance

One member of our team (Yangfan Yu) did the human test, we choose a web simulator for Atari games, and we select the gray-image model to be same with what DQN actually deals with.

Figure 2: Example of Human Player Game Screen

After half day's training for each game, the member started to play each game for 50 times. And we take the average of the score for each game as human performance.

Game	Average Score(50 games)
Freeway-v0	19.83(Keyboard)
Breakout-v0	26.60(Keyboard)
Pong-v0	10.27(Mouse)

Table 2: Human Performance

4.2 Random Policy

We test the three games without prior training, which are all played by a random policy agent. The random agent can easily be stuck and keep actions back and forth or not move.

Game	Average Score(50 games)
Freeway-v0	0.7
Breakout-v0	1.28
Pong-v0	0

Table 3: Random Agent Performance

4.3 DQN with uniformly random sampling

Our goal of the training is the DQN agent can perform greatly better than random agent and be close to human performance or even do better. We simultaneously trained the three games on google compute engine, each for five days, and each for about saven million of time steps. After several bad attempts, , we set the hyperparameters based on our experience and others' experience and research outcomes as below:

Hyperparameter	Value
FRAME_PER_ACTION	1
GAMMA (decay rate of past observations)	0.95
OBSERVE (timesteps to observe before training)	5000
EXPLORE (frames over which to anneal epsilon)	1000000
FINAL_EPSILON	0.1
INITIAL_EPSILON	1
REPLAY_MEMORY	40000
BATCH_SIZE	32
UPDATE_TIME	10000

Table 4: Hyperparameter Values

After training, we test the DQN agent for each game 50 times respectively.

Game	Average Score
Freeway-v0	18.60
Breakout-v0	24.55
Pong-v0	14.37

Table 5: DQN Agent Performance

The DQN's performance on Freeway and Breakout got closer to the human tester's performance, and it did better than the human tester.

4.4 Summary

We can see that DQN requires quite much time to train to get close to human player level, and it needs more time to exceed human player. Pong-v0, DQN can do better than our tester, but that might partly due to the control method if game, it seems using mouse to control the Pong bat is more difficult than keyboard for human player on this game, to DQN agent, the control method does not matter.

To improve our DQN agent, apart from training more time, there is still much work we can do like modifying the parameters, trying other exploration methods such as Softmax, and using other sampling approaches like Gibbs sampling.

Actually, after DeepMind's DQN, deep reinforcement learning has made a lot of improvements, like A3C based on Actor-Critic framework and UNREAL based on A3C can greatly increase the efficiency and outcome of games.

5 Acknowledgements

During the project, Xiaoming Wu learned and implemented the DQN algorithm and wrote the Motivation and Related Work part.

Yangfan Yu did the human performance test and random agent test for the three games and wrote the Approach and Evaluation part.

Both of us read related papers and contributed to the setting of hyperparameters and the implementation of the algorithm.

Both of us did training work and evaluation of DQN agent.

We would like to thank DeepMind for their research on DQN, OpenAI for their providing of Gym-Atari environment, Retrogames for its Atari simulator for human players, Google for its cloud computing service and python community for Anaconda.

References

[Rummery G A, Niranjan M, 1994] Rummery G A, Niranjan M. Online Q-learning using connectionist systems[M]. University of Cambridge, Department of Engineering, 1994.

[Watkins, et al., 1992], Watkins, Christopher JCH, and Peter Dayan. Q-learning[J]. Machine learning, 1992, 8(3-4): 279-292.

[Barto A G, 1995] Barto A G. Adaptive Critics and the Basal Ganglia[J]. *Models of information processing in the basal ganglia*, 1995: 215.

[Richard Bellman, 1957] Richard Bellman. A markovian decision process. *Technical report, DTIC Document,* 1957.

[Levine, John, et al., 2013] Levine, John and Bates Congdon, Clare and Ebner, Marc and Kendall, Graham and Lucas, Simon M. and Miikkulainen, Risto and Schaul, Tom and Thompson, Tommy. General video game playing[J]. *Dagstuhl Follow-Ups*: 77-84, 2013. Dagstuhl Publishing, pp.

[Perez-Liebana D, Samothrakis S, Togelius J, et al, 2016]. General video game ai: Competition, challenges and opportunities[C]. *Thirtieth AAAI Conference on Artificial Intelligence*. 2016: 4335-4337.

[Mnih V, Kavukcuoglu K, Silver D, et al., 2015] Human-level control through deep reinforcement learning[J]. *Nature*, 2015, 518(7540): 529.

[Mnih V, Kavukcuoglu K, Silver D, et al., 2013] Playing atari with deep reinforcement learning[J]. *arXiv preprint arXiv*:1312.5602, 2013.]

[Brockman G, Cheung V, Pettersson L, et al., 2016] Openai gym[J]. *arXiv preprint arXiv*:1606.01540, 2016.

[Defazio A, Graepel T, 2014]. A comparison of learning algorithms on the arcade learning environment[J]. *arXiv preprint arXiv*:1410.8620, 2014.

[Bellemare M G, Naddaf Y, Veness J, et al., 2013] The Arcade Learning Environment: An evaluation platform for general agents[J]. *J. Artif. Intell. Res.(JAIR)*, 2013, 47: 253-279.