# **Deep Reinforcement Learning on OpenAI Gym Games**

|  |  |
| --- | --- |
| Lingjie Kong1  Stanford University  Stanford Center for Professional Development  ljkong@stanford.edu | Ruixuan Ren1  Stanford University  Stanford Center for Professional Development  ruixuan@stanford.edu |

**Abstract**

*Reinforcement Learning (RL) is an area regarding how agents act to an environment for maximizing its rewards. Unlike Markov Decision Process (MDP) in which agent has full knowledge of its state, rewards, and transitional probability, reinforcement utilizes exploration and exploitation to cover model uncertainty. Due to the model usually has a large input feature space, a neural network (NN) is often used to summarize the correlation between input feature and output state action value. Our goal is to improve existing algorithms or potentially develop new algorithms, specifically double A3C. We will implement DQN, double DQN, dueling DQN and A3C (Asynchronous Advantage Actor-Critic) to play OpenAI Gym Atari 2600 games to obtain benchmark performance. Then we will propose our implementation on double A3C, an improved version of state-of-the-art A3C algorithm. We will compare its performance, data efficiency and computation efficiency to the other methods.*

# **Introduction**

Behaviorist psychology regarding taking the best actions to optimize agent’s reward at a specific state inspired the development of reinforcement learning. Up to now, reinforcement learning has been studies in many disciplines such as control theory, information theory, statistics, and so on.

Markov Decision Process (MDP) was used to solve classical decision-making problem where agent has full knowledge of the environment including state, reward, and transitional probability. Due to the limitation of full knowledge of the environment, Q learning was developed to let agent explore to find potential optimal solution as well as exploit to optimize the current good solution.

Due to large input state space, it is impossible to use a look up tale like MDP, neural network is used instead. Under the condition that neural network is universal function approximator, it can capture the non-linear relationship between input and output. Neural network will be trained by using forward and backward propagation to train the weight at each hidden layer. With fully trained model, it will be used to inference based on the current state input, what will be the optimal action to take in order to maximize its rewards.

Reinforcement Learning bring new challenges on how to build and train an efficient neural network. Specifically, Reinforcement Learning must learn from sparse and noisy data the agent collected through its interaction with the environment. These sparse and noisy data might cause instability during learning. Moreover, reward is usually delayed until the end of episode. Therefore, it requires efficient method to propagate the reward to prize the early good actions during training. Fixed underlying distribution is an assumption for neural network model. However, as the agent interact with the real environment, it might change the environment. Therefore, Reinforcement learning is also required to capture the dynamic of environment as well.

The bellows are the details that will paper will cover. First, it presents Convolutional Neural Networks (CNN) to explain how features will be extracted from each frame of the game. Then, this paper will review classical DQN to train neural networks with Adam optimizer. Then, this paper will explain why double DQN outperform classical one. Next, this paper will explain how dueling DQN as well as A3C is a better network structure. Finally, this paper will present Double A3C which utilize the strength from both double QND as well as A3C.

We compare our result on three Atari games: Pong, Breakout, and Ice hockey.



Figure (left to right) Pong, Breakout, Ice Hockey

# **Related Work**

Figuring out how to control agent from high-dimensional inputs like vision input is one of the biggest challenges of reinforcement learning. Most successfully RL model before is based on carefully selected feature with linear combined values. Obviously, the quality of the selected feature representation will largely influence the performance.

With the fast development of computer vision, it leads to some breakthroughs on how to extract the feature representation more efficiently by using more efficient models [1]. All these methods utilize ideas of neural network structures such as Convolutional Neural Networks, (CNN), Recurrent Neural Networks (RNN), Multilayer Perception, Boltzmann Machine Graphic Model, and so on.

Besides the challenge from the input feature representation, reinforcement learning presents other challenges. First, traditional machine learning requires large number of carefully labelled data. However, reinforcement learning algorithm has to learn from scalar rewards which is most of the time noisy and delayed from the current state. Second, unlike most supervised learning algorithm which assume the independence of samples, RL’s sample are highly correlated.

Q-Learning algorithm [2] with stochastic gradient descent is often used to train reinforcement learning model. In Q-Learning algorithm, we need to store and update a Q value estimate Q(*s, a*) for each (*s, a*) pair, where Q(*s, a*) is the expected utility or value of taking action *a* in state *s* and then following the optimal policy afterwards. However, if we have a large state or action space, it will be expensive to store Q values for all (*s, a*) pairs. One of the common solutions to this issue is to use function approximation, where we extract featuresfrom (*s, a*) and define a functionto approximate Q(*s, a*). Then optimizing the estimation of Q values turns into optimizing the parameters in.

Deep Reinforcement Learning [3] uses a deep neural network, which is called Deep Q-Network (DQN), as the approximate functionof Q values. This research showed that the agents trained by DQN can achieve high performances in playing Atari 2600 games in most cases. Further studies of Double DQN [4] and Dueling DQN [5] proposed methods to improve the convergence speed and final performance of DQN. All the DQN models mentioned above can be trained with GPU at a high speed.

Recently, asynchronous method has been proposed to apply to the Deep Reinforcement Learning [6]. The study showed that their best algorithm, Asynchronous Advantage Actor-Critic (A3C), can be trained 2x faster than DQN even if it uses a multi-core CPU instead of GPU. Moreover, agents trained by A3C can achieve higher performances in most of the Atari 2600 games than DQN models.

# **Approach**

First, the input to the network is each frame of the game in one episode. Each frame was past into convolutional neural network. The general structure of convolutional neural networks including convolutional layers, maxpooling layers, activation functions, and fully connected layers. The general structure of a convolutional neural network refer to AlexNet [1] as below.



Figure Convolutional Neural Networks

Classical DQN will pass casted convolutional layer values into several fully connected layers and eventually generate outputs which has the same dimension of all possible actions. Which action has the higher value will be the optimal action in that specific state.

DQN is usually trained with experience replay which fully utilizing the limited amount of data we have for RL. Basically, the agent will store its previous transition and sample to update its Q value. DQN will be trained by using SGD to minimize the L2 loss between the current Q value and the discounted next Q value by taking the best action plus reward. The overall algorithm [3] is as below.



Figure DQN

However, classical DQN algorithm showed above may suffer from overestimation, and this problem can be improved by utilizing ideas behind the Double Q-learning algorithm. In Double Q-learning, two value QA and QB functions are learned, with one to determine the greedy policy and the other to determine its value [7].

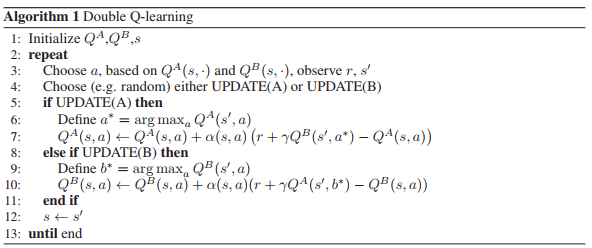


Figure Double Q Learning

Double DQN, in comparison to Double Q-learning, replaces the weights of the second network with the weights of the target network for evaluation of the current greedy policy, thus eliminating the need of an additional network. It is shown that Double DQN finds better policies and obtains better results on the Atari 2600 domain [4].

The L2 loss will be shown as below which will be taken gradient to update it Q value. In the double DQN equation, w will be update each iteration while will only be assigned by w at the end of each episode.

Another improvement to classical DQN is the Dueling Network Architecture [5]. Compared to the single stream Q-network, the dueling network has two streams to separately estimate state value function and the state-dependent action advantage function. It is demonstrated that the dueling architecture can more quickly identify the correct action during policy evaluation as redundant or similar actions are added to the learning problem. The reason that Dueling Network works better because it separates the value function from its action advantage function which allow the network to capture each individual one better.



Figure 5 Single stream Q-network (top) and dueling Q-network (bottom)

Different previous method, dueling Q-network define the network by its state’s value as well as its advantage function for a specific action at a state where and are the parameters of the to streams of fully-connected layers.

The algorithm that has demonstrated the best performances in Atari 2600 domains is asynchronous advantage actor-critic (A3C) algorithm. A3c is a multi-threaded asynchronous variant of advantage actor-critic algorithm, where the actor aims at improving the current policy and the critic evaluates the current policy. Like the dueling network architecture, A3C also implements a network which contains two streams to separately update the parameters of the policy and parameters of the value function. The algorithm of A3C is as below [6].



Figure 6 A3C

Our approach will be based on the Double Q learning [7], Double DQN [4], and state-of-the-art A3C algorithm [6]. We call it double A3C. The key technique in Double DQN is to use one deep neural networks with the same structure but different instance as target network and trained network. Meanwhile, double Q-learning has different parameters as two approximate functions of Q values. Both techniques can reduce the overestimations of action values under certain conditions and improve the agent performance.

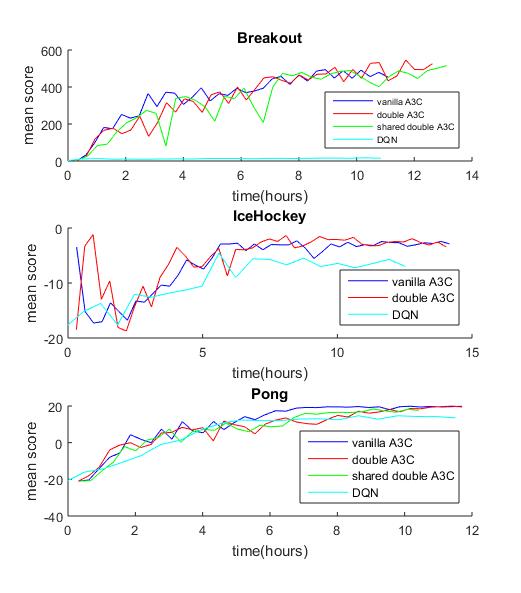
We believe the same technique can also be applied to A3C algorithm. Under the condition that double DQN concept has already been implement as target network during update. We are wondering whether we can add double Q-learning on top of that to further speed up the convergence.

To be more concrete, we add another set of value parameter and random pick from one set to update both and , we hope that by having two value parameters inspired by double Q learning, it can break out the correlation in each sequence of sampled data which leads to faster convergence.

# **Experiment**

We train and evaluate our approach using the environment of OpenAI Gym Atari 2600 games. Three games are chosen: Breakout, Ice Hockey, Pong. We compare the average performance of agents trained by Vanilla A3C, Double A3C, Shared Double A3C, DQN to measure success. Moreover, we analyze whether our proposed method will benefit or harm the convergence speed of A3C algorithm.

Figure 7 shows how the average total reward evolves during training on three games when using Vanilla A3C, Double A3C, Shared Double A3C and DQN methods. In Breakout game, three different A3C methods yield dramatically higher mean scores than DQN within the same amount of time of training. In Ice Hockey game, all A3C methods still lead to relatively better mean scores than DQN. In Pong game, all four methods lead to similar results.



*Figure 7 Average performance and training speed comparison of three different A3C methods and DQN on three Atari games. The x-axis shows the time in hours. The y-axis shows the average score.*

We would also like to compare data efficiency among three different A3C methods for future work. We also like to try policy search as well as other algorithms to compare the performance.

# **References**

[1] A. Krizhevsky, I. Sutskever, and H. Geoffrey E., “ImageNet Classification with Deep Convolutional Neural Networks,” *Adv. Neural Inf. Process. Syst. 25*, pp. 1–9, 2012.

[2] C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Mach. Learn.*, vol. 8, no. 3–4, pp. 279–292, 1992.

[3] D. Zoran *et al.*, “Playing Atari with Deep Reinforcement Learning,” *arXiv*, vol. 32, no. Ijcai, pp. 1–9, 2016.

[4] D. S. H. Van Hasselt, A. Guez, “Deep Reinforcement Learning with Double Q-learning,” p. in proceedings of AAAI 2016.

[5] Z. Wang, N. de Freitas, and M. Lanctot, “Dueling Network Architectures for Deep Reinforcement Learning,” *arXiv*, no. 9, pp. 1–16, 2016.

[6] V. Mnih et al., “Asynchronous Methods for Deep Reinforcement Learning,” p. in proceedings of ICML 2016.

[7] H. Van Hasselt, A. C. Group, and C. Wiskunde, “Double Q-learning,” *Nips*, pp. 1–9, 2010.