Deep Learning Techniques in Reinforcement Learning

Lingfei Li (senior undergrad in CS)
Shuo Niu (2nd yr grad in Statistics)
Tong Niu (2nd yr grad in Statistics)
Weifeng Zhu (2nd yr grad in Statistics)

ABSTRACT

In this paper, we examine the performances of several popular reinforcement learning methods on video games and study the advantages of applying deep learning techniques to them. We implemented DQN, Policy Gradient and Actor Critic algorithms with the techniques of convolutional layers, LSTM cells, dropout regularization and batch normalization, whose input is raw pixels of the game screen. We investigate the performance of the algorithms on three games, including CartPole, Pong and SpaceInvaders. We find that Actor Critic algorithm offers the best result among all RL methods, and deep learning techniques can effectively improve the performance of most methods.

INTRODUCTION

Reinforcement learning is a type of machine learning that allows an autonomous agent to improve its performance by acting and receiving "reward" signals. The goal of reinforcement learning is to learn good policies for sequential decision problems. This automated learning scheme implies that there is little need for a human expert and much less time will be spent designing a solution. Another advantage of RL is that a large number of Artificial Intelligence problems can be mapped to a sequential decision process, making the applications of RL quite abundant.

Over the last decades, advances in both machine learning algorithms and computer hardware have led to more efficient ways for training deep neural networks. Deep learning techniques, such as CNN, have shown great potential in feature representations. By incorporating existing deep neural network architectures into RL methods, we can operate directly on RGB images without manual feature engineering and efficiently train data by stochastic gradient updates. In other words, we obtain deep reinforcement learning methods when we use deep neural networks to approximate any of the following components of RL: value function, policy or model. In recent years, deep reinforcement learning has achieved promising performance. In 2015, A deep Q-network implemented by Google DeepMind outperformed almost all of other state-of-the-art AI controllers and 75% human players on 49 Atari-2600 games [1]. Their recent work on asynchronous RL methods has obtained even faster training speed and more advanced agent performance [5].

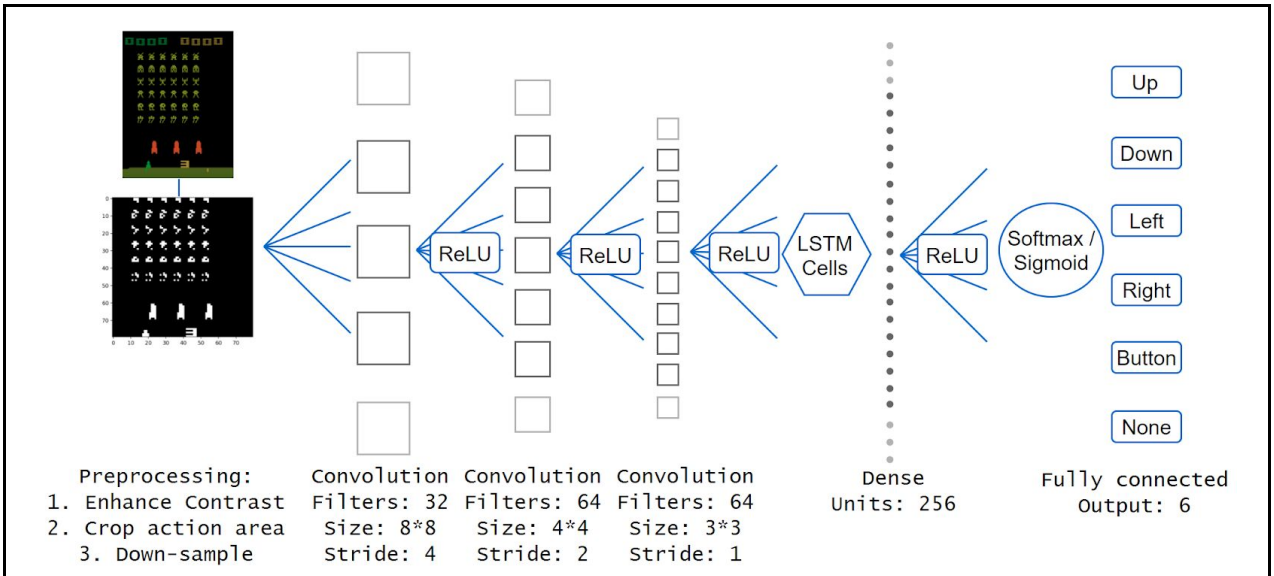

Figure 1. The structure of our network with 3 convolutional layers and LSTM cells. In some experiments, some parts of the network is removed to compare the performance.

This paper presents our experiments with several deep reinforcement learning methods used to learn to play two Atari 2600 games, Pong and Space Invader. Atari 2600 (included in OpenAI Gym) is a RL testbed that has agents with a high dimensional visual input (210 * 160 RGB video at 60HZ) and a set of tasks that can be challenging for human players. Our goal in this experiment is to (1)examine the performances of different RL methods, including deep Q-Network [2], Policy Gradient [4] and Actor Critic [5], and (2)study the improvement of these methods when incorporated with deep learning techniques like convolutional layers and LSTM cells.

From our experiment results, we confirm that RL can always benefit from CNN and RNN [3] to get more reward and we will show that Actor Critic performs better than DQN in our trials. In the end, we also come up with ideas to improve the performance in future work.

PROBLEM DEFINITION AND ALGORITHM

1. Task Definition

In this paper, we selected three problems to conduct our experiments: a basic RL problem CartPole and two Atari 2600 video games Pong and Space Invaders.

The CartPole game provides 4 real numbers as the state (cart position, cart velocity, pole angle, pole tip velocity), and receives 2 possible actions (applying a unit of force to left or right). When the cart is off the starting point by 2.4 units, or the pole angle is larger than 20.9 degrees, the game is over and the problem is considered failed. If the game lasts for 200 steps, the problem is considered solved. For each step in the game, a +1 reward is

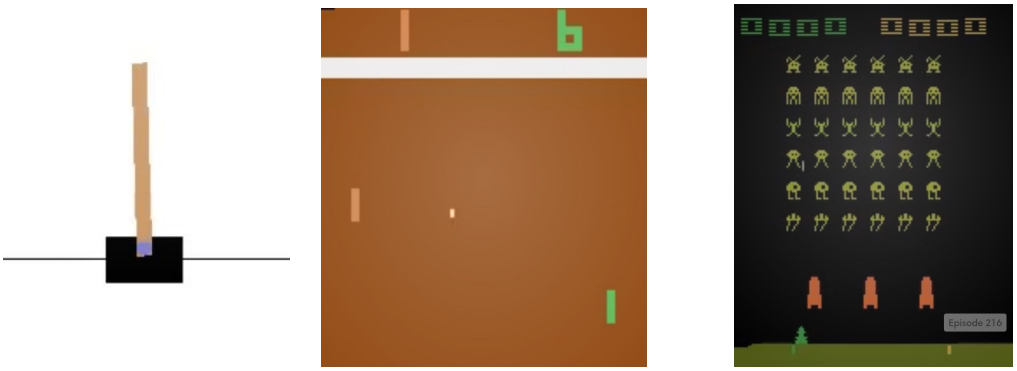

Figure 2. Screenshots of the games (from left to right): CartPole, Pong, Space Invaders

offered to the agent. When the game is over, no reward or penalty is offered by default, but we found that this value has significant impact on the learning performance.

For Atari games, our networks used preprocessed raw pixels as input. Specifically, the game screen is a series of 210*160 images in RGB channels (as shown in Figure 2). To enhance contrast and avoid useless graphical information, we removed the background by setting its pixel values to 0, and converted all other pixels values to 1. To reduce computational complexity, we extracted the central 160*160 pixels of the images (approximately the action area) and down-sampled it to 80*80. Hence, the input fed to the network is an 80*80 bitmap. The output of our networks is 6 real numbers that correspond with the 6 actions available in Atari games (4 directions, “fire” and “no action”). Both Atari games use the raw game score as the reward to the agent.

For each RL algorithm, we conducted experiments with 3 different network structures. We first tested single hidden layer network with 256 hidden units. Then we added 2 convolutional layers. The first convolutional layer has 32 filters with size 8*8 and stride 4. The second convolutional layer has 64 filters with size 4*4 and stride 2. This is the standard configuration that DeepMind used in their DQN paper [2]. In the third network we added another convolutional layer that has 64 filters with size 3*3 and stride 1. Our choices of hyperparameters in these networks mostly followed DeepMind’s DQN paper. For an illustration of our network structures, please see Figure 1.

2. Algorithm Definition

Our experiments are mostly based on Q-learning, a model-free temporal-difference RL method that learns the action-value function and chooses the action giving the highest value.

Deep Q-Network (DQN) is an enhanced version of Q-learning with three major changes. It uses (1)deep network (CNN) to deal with raw pixel input, (2)experience replay for minibatch updating to break correlation in training data and (3)two sets of parameters to make the updating process more stable [1]. Deep Recurrent Q-Network(DRQN) uses recurrent neural network (RNN) with the technique of Long Short-Term Memory(LSTM) to learn more information in earlier states. It can significantly outperform traditional methods in partially

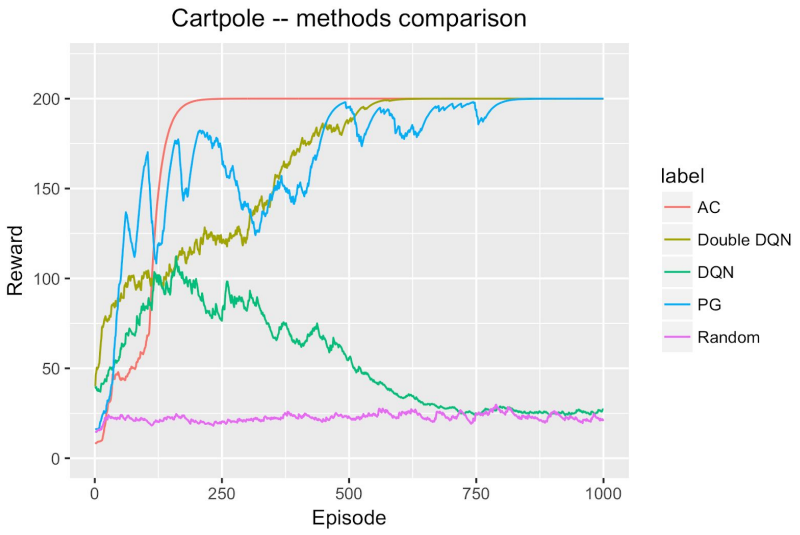

Figure 3. Performance of four methods used in CartPole with random player as baseline. The lines represent the moving average reward of each algorithm.

observable problems, where it is hard to make decision based on limited current state information.

Some of our experiments are also based on Policy Gradient (PG) method. Unlike Q-learning that finds the optimal policy by calculating a value function, PG directly learns the policy, which is the probability distribution of actions. PG has better convergence property than Q-learning, but it can be inefficient to train and may get to local minimum.

Actor-Critic (AC) algorithm aims to benefit from the ensemble of PG and DQN. Given a state, the Actor (a PG agent) chooses an action using its own parameters, while the Critic (a DQN agent) estimates the action-value function for this choice, and passes it to Actor to adjust its parameters. This method can reduce variance effectively, and we suppose it can achieve a better performance than other methods.

EXPERIMENTAL EVALUATION

1. Methodology

In our experiments, we used the OpenAI Gym as our simulator and wrote our codes in Python/Tensorflow. We trained the agents on an AWS EC2 p2.xlarge instance that supports GPU acceleration using an NVIDIA K80 graphic card. Each agent was trained approximately 20 hours until its performance stabilized.

To measure the performance of the RL agents, we followed the evaluation strategy introduced by Bellemare et al. (2013) [6]. Specifically, during the training process, we recorded the total

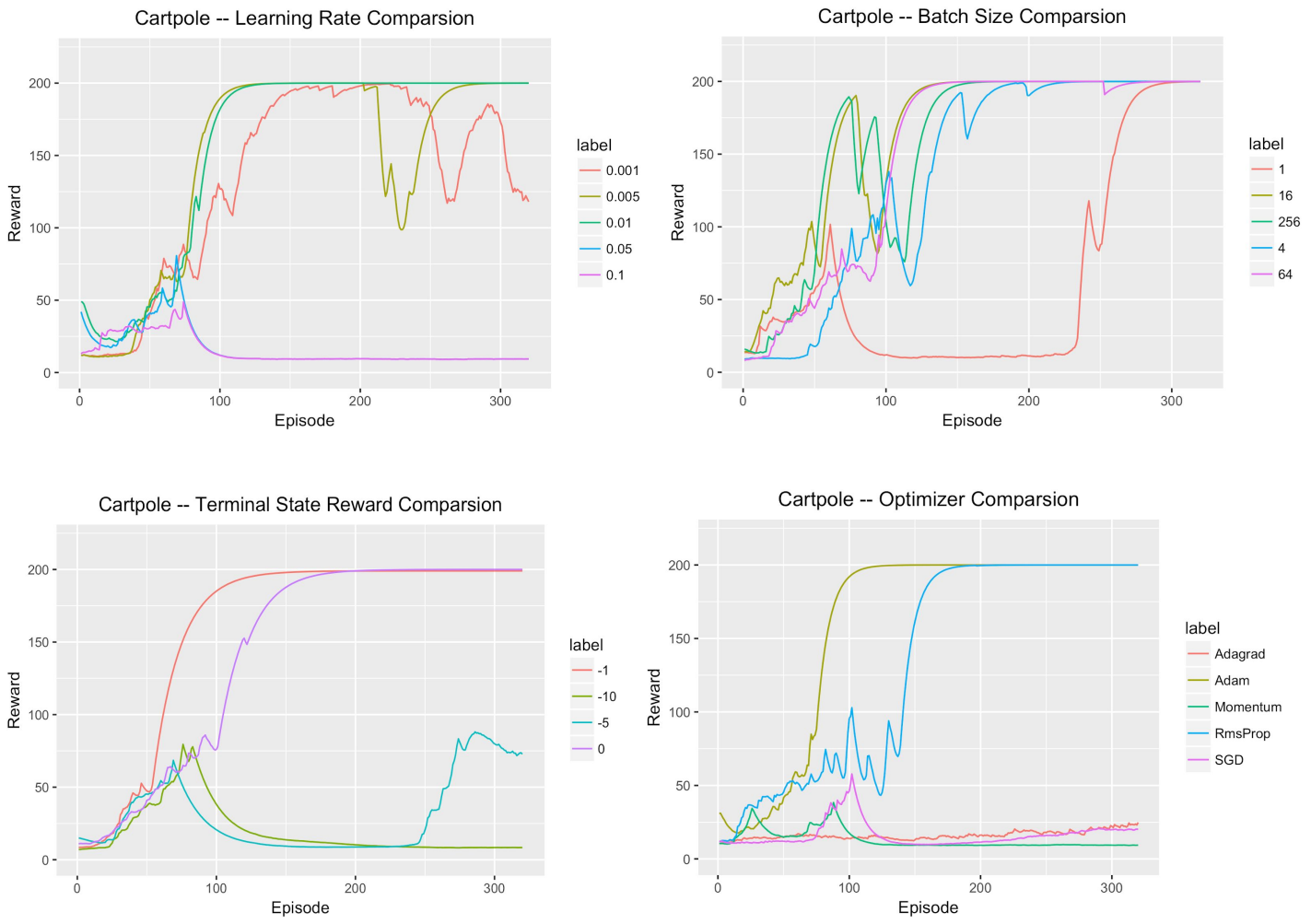

Figure 4. Comparison of the performance of AC used in CartPole with regard to different batch sizes, learning rate, optimizer and terminal state values, respectively. The lines represent the moving average reward of each algorithm.

rewards that an RL agent obtained in each episode. When the training process completes, we calculated moving average rewards for each agent and used this data to plot learning curves. In some alternative performance measurements, Q value and loss are also recorded and plotted. In our results, we didn't include them for 2 reasons. Firstly, we used different RL algorithms (DQN, PG, AC) that have different definition of value function and loss function, thus the comparison across agents would not be meaningful. Secondly, Q value and loss are not convincing performance indicators. Due to the divergence property of Q learning, it is possible that an agent constantly calculates a large Q value and tiny loss for a certain action

in all situations. This will guide the agent to choose the same action during the entire game, which almost always leads to the worst performance.

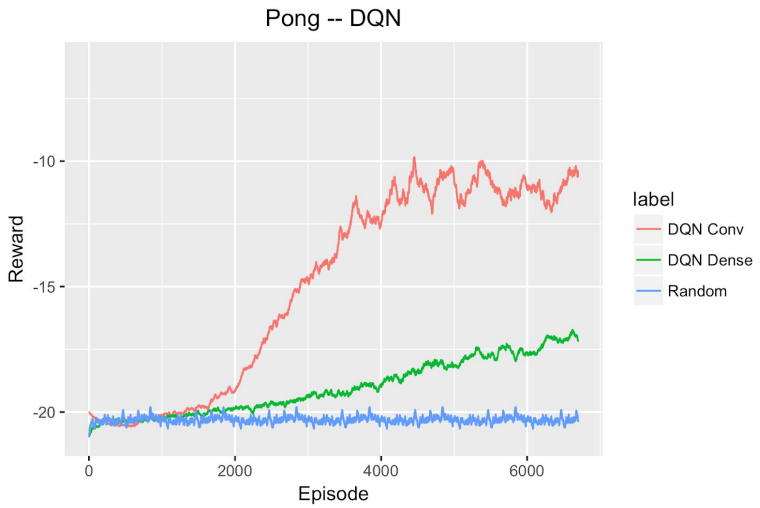

Figure 5. Performance of DQN with and without convolutional layers in Pong using random player as baseline. The lines represent the moving average reward of each algorithm.

2. Results

a. CartPole

We tried four algorithms in CartPole for comparison and the result is shown in figure 3. DQN offers the worst result in this game. It grows slowly in the beginning and drops to the level of random player in the end due to its unstable convergence property. After applying the Double DQN technique [9], it can successfully learn the game, which is possibly a result from correlation breaking in training data. PG converges fast at the beginning, but is more prone to performance oscillation. As an ensemble of DQN and PG, AC both converges fast and staliblizes in performance in the experiment.

So we used AC to train more models with different batch sizes, learning rate, optimizer and terminal state values. Models without minibatch converge slowly and for other different sizes shown in figure 4 they perform similar in this game. For learning rate, 0.005 and 0.01 work best, while 0.1 is too large and fails learning completley. Among all optimizers, Adam performs best in our experiments. First three tricks are always seen when training neural networks, while we also found that adding some penalty in terminal state of each episode helps and -1, -10 are good choices.

b. Pong

As an Atari game, Pong uses raw vision input to the model and has a much slower learning process than CartPole. Due to the insufficiency of budget and time, we only used DQN to test the effects of applying convolutional layers in this experiment.

From the results displayed in Figure 5, we can see that playing Pong randomly is always a "lose" strategy (-21 reward means winning no games in a round), and DQN without convolutional layers can only do slightly better. With the power of convolutional layers, our

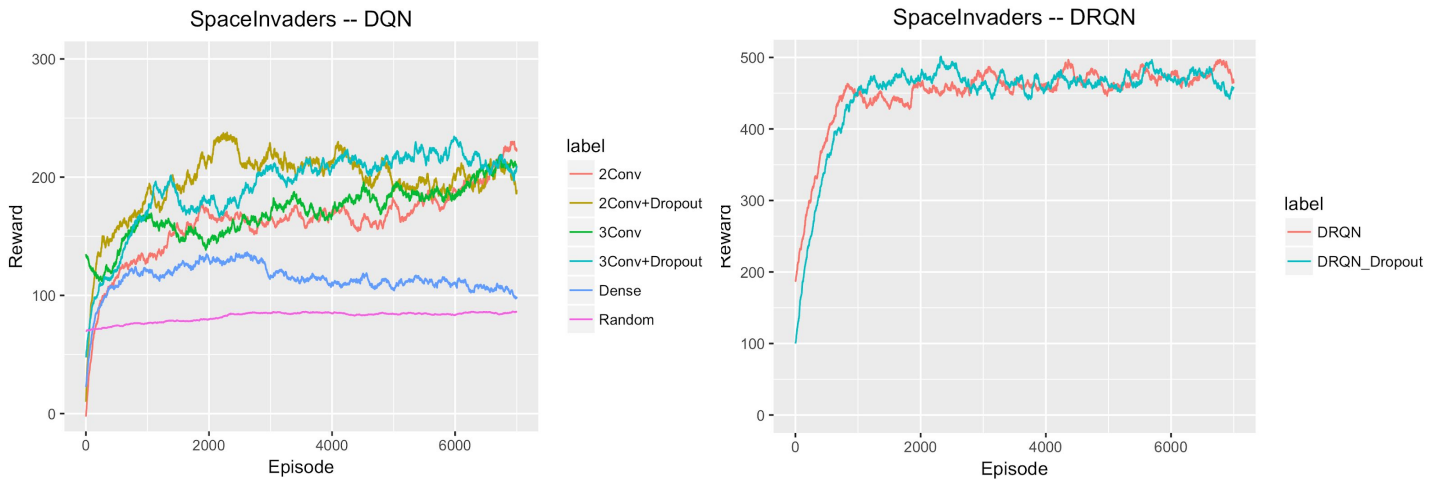

Figure 6. Performance of different models settings in DQN for Space-Invaders. The lines represent the moving average reward of each algorithm.

DQN model can achieve a more satisfactory intelligence level while still has potential for further improvement.

c. Space Invaders

For Space Invaders, we tried two RL methods (DQN, AC) with several deep learning techniques (convolution, LSTM, dropout).

The results of experiments using DQN are shown in Figure 6. From the results we can see that DQN with deep learning techniques always outperform DQN Dense (Q-learning without any deep learning technique). Among different settings of DQN with deep techniques, the learning performance is pretty similar. Dropout helps a lot in both 2-conv and 3-conv models, which makes training process faster and more stable. But we did not expect that 3-conv models have similar performance to 2-conv ones. We suppose that this phenomenon is due to the difficulty to train deeper neural network.

We choose 2-conv to do a DRQN training shown in the left, and the results are much better than before. Both results converge much faster and reach a higher result than before, profiting from more information bringing by RNN.

Actor Critic is the best in all algorithms as we expected, whose reward can reach 600 presented in Figure 7. In the Critic part, like in DQN, we tried different settings for CNN and

3-conv+Dropout is the best one. However, batch-normalization does not act well in this experiment.

To summarize the performances across different RL methods, in figure 7, we also plotted the learning curves for each method with the model setting that works best for it. From the graph we can clearly see that AC methods have the overall best performance among all three methods, which supports and confirms our hypothesis.

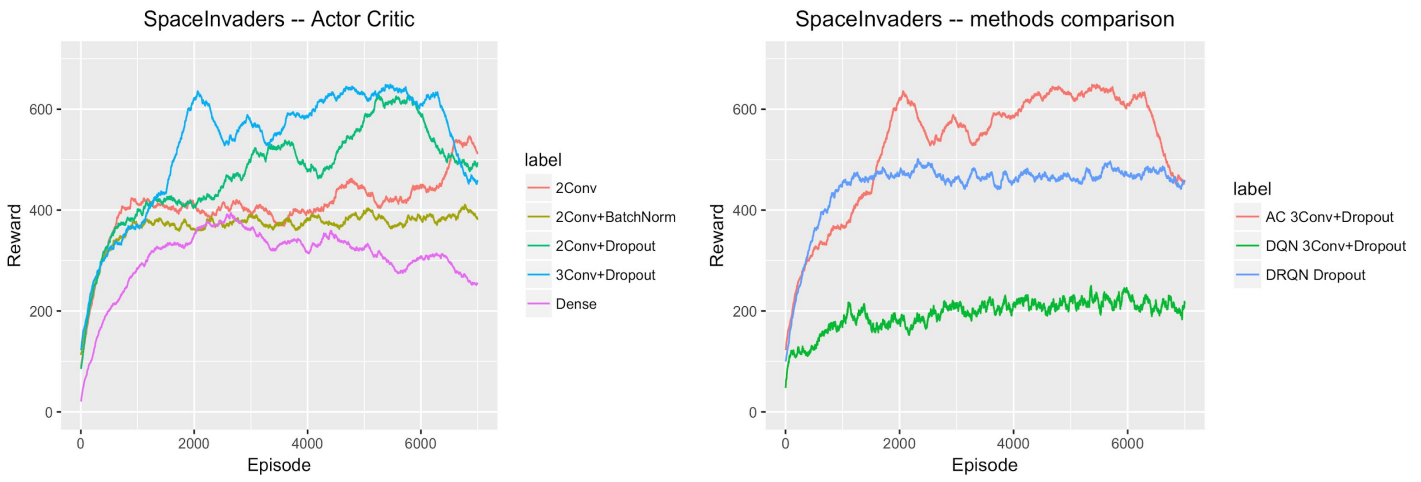

Figure 7. The left figure shows the performance of different model settings of Actor Critic for SpaceInvaders. The right figure compares the performance of the best curve in each of the three main methods: AC, DQN and DRQN for SpaceInvaders. The lines represent the moving average reward of each algorithm.

DISCUSSION

One of our hypotheses is that by applying deep learning technique such as convolutional layer and LSTM, we achieve a better performance. This matches our result. CNN has an advantage of well analyzing raw video data while LSTM combines the information of previous video frame and current state.

In addition, the Actor-Critic outperforms DQN as we expected. This result supports the strength of Actor-Critic that it has better convergence properties than value-based reinforcement learning methods benefiting from the Actor part which employs policy gradient update.

Furthermore, we expect dropout regularizations can improve the performance of the deep neural network, which is confirmed by our result. Dropout constrains the network adaptation to data at training and thus prevents overfitting.

Actually, some curves of Actor Critic drop after reaching a high value, which is a little bit confusing. There might be two reasons leading to this. First is that Q learning is hard to learn though it can reach the global optimum, since too much exploration is needed and better

training policy is needed here. The other one can be overestimations [7] in critic part, which brings noise.

INDIVIDUAL CONTRIBUTIONS

Lingfei Li:

- Algorithm Implementation (DQN, PG)
- Experiments (DQN, PG)
- Environment setup

Shuo Niu:

- Algorithm Implementation (PG, AC)
- Experiments (AC)
- Result Analysis and Visualization

Tong Niu:

- Experiments (DQN, PG)
- Result Analysis and Visualization
- Literature Review

Weifeng Zhu:

- Algorithm Inplementation (DRQN, AC)
- Experiments (DRQN)
- Literature Review

RELATED WORK

OpenAI has developed an open source toolkit (Gym) for developing and comparing reinforcement learning algorithms. Thanks to their efforts, we can easily manipulate environments for hundreds of different games and compare our agents with others on their website.

For reinforcement learning, DeepMind has made great exploration and contributions so far. In terms of methodology, their research provides most of the state of art algorithms such as

Double DQN, policy gradient, actor critic,etc. Besides, they also successfully developed powerful agents for various games including the amazing AlphaGo.

FUTURE WORK

There are two major shortcomings for our agent. First, the performance of agents doesn't meet our expectation because of unstable rewards between different episodes and being uncompetitive in comparison with human players. Another problem is unsatisfactory training efficiency. Currently, we only trained around 7000 episodes for each agent on AWS p2.xlarge instances, which already used up our AWS educational credits. Due to the large number of agents to be trained and the limitation of budgets, we didn't continue the training process.

To improve the RL agent performance in the future, we plan to use region based CNN to estimate the value function along with the Asynchronous Advantage Actor-Critic (A3C) algorithm to boost the agents' capability. To speed up the training process, our future plan is to implement paralleled computation with multiple GPU at the same time and use more advanced AWS EC2 instances.

CONCLUSION

In this project, we implemented two major reinforcement learning algorithms with various value approximation functions to train agents for different games. According to experiments results, we compare the difference between Deep Q Networks and Actor Critic RL algorithms, evaluate performance of different neural network structures like dense, CNN and RNN and arrive at the following conclusions.

Firstly, Actor Critic algorithm outperforms traditional Deep Q Network under most neural network structures in all games. For example, in Space Invaders, Actor Critic agents can achieve average rewards of 600 when tested with CNN, while the DQN agents can only achieve around 200. Secondly, CNN can significantly enhance the agent's intelligence in comparison with fully connected neural networks. Nevertheless, usage of RNN for value function approximation doesn't receive an ideal improvement. Finally, under our current computation power and settings of agents' algorithms, we can't get an agent to behave closely to humans. Overall, these findings have supported our hypotheses of the advantages of applying deep learning techniques in reinforcement learning.

BIBLIOGRAPHY

[1] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Petersen, S. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529-533.

[2] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

[3] Hausknecht, M., & Stone, P. (2015). Deep recurrent q-learning for partially observable mdps. *arXiv preprint arXiv:1507.06527*.

[4] Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)* (pp. 387-395).

[5] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., ... & Kavukcuoglu, K. (2016, June). Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning* (pp. 1928-1937).

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

[7] Van Hasselt, H., Guez, A., & Silver, D. (2016, March). Deep Reinforcement Learning with Double Q-Learning. In *AAAI* (pp. 2094-2100).

[8] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W., (2016). OpenAI Gym. *Computing Research Repository, abs/1606.01540.*

[9] Hasselt, H., Guez, A., & Silver, D., (2015). Deep Reinforcement Learning with Double Q-learning. *Computer Research Repository, abs/1509.06461*