2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
Banff Center, Banff, Canada, October 5-8, 2017

Effective Lazy Training Method for Deep Q-Network in Obstacle Avoidance and Path Planning

Juan Wu[1], Seabyuk Shin[1], Cheong-Gil Kim[2], Shin-Dug Kim[1]
Department of Computer Science
[1]Yonsei University, [2]Namseoul University
Seoul, Korea
wujuan1117@hotmail.com, dawnshin2000@naver.com, cgkim@nsu.ac.kr, sdkim@yonsei.ac.kr

Abstract— **Deep reinforcement learning technique combines reinforcement learning and neural network for various applications. This paper is to propose an effective lazy training method for deep reinforcement learning, especially for deep Q-network combining neural network with Q-learning to be used for the obstacle avoidance and path planning applications. The proposed method can reduce the overall training time by designing a lazy learning method and a method removing unnecessary repetitions in the training step. These two methods can reduce a significant portion of total execution time without losing any required accuracy. The proposed method is evaluated for the obstacle avoidance and path planning tasks, where an agent trapped in an unknown environment is trying to find out the shortest path to the destination without any collision, through its self-study. And the experiment results show that the proposed method reduces 53.38% of training time on average, compared to the traditional method with no performance loss and make the training procedure more stable.**

Keywords—Deep Reinforcement Learning, Deep Q Network, Neural Network, Path planning

1. INTRODUCTION

As the machine learning and the artificial intelligence turn into focus, deep reinforcement learning has become popular in many challenging areas, such as unmanned aerial vehicle (UAV), autonomous vehicles, playing Atari games, robot control, and so on. Deep reinforcement learning combines reinforcement learning and neural network and forms a new research hotspot in the field of artificial intelligence. The basic idea of reinforcement learning is to learn an optimal strategy to maximize the cumulative reward obtained by the agent from the environment [1]. At first, the agent knows nothing about the environment. However, through its self-study, the agent learns from the mistakes and then it can find the best way to achieve the goals. And through the neural network, we can extract the features easily and automatically from high-dimensional data.

One kind of deep reinforcement learning is deep Q-network [2], which combines Q-learning and neural network. That is we can put the current state and action as the input of the neural network, then through the analysis of the neural network, we can get the action's Q-values. Or we can only enter the state and get all action's Q-values through the neural network, then we can select the maximum Q-value action. As we can see, we do not have to record the Q-values in the table, but just directly through the neural network, the Q-value can be generated. We can imagine that the neural network receives external information in a similar way to the human eyes and nose. And then through the brain processing, the value of each action is generated. Finally, any particular action is selected by reinforcement learning method. Deep Q-network has successfully applied in many domains such as unmanned aerial vehicle [3], autonomous vehicles [4], and many other areas.

In this paper, we have proposed an effective lazy training method for deep reinforcement learning, especially for deep Q-network, which combines neural network with Q-learning. The specific method is designed as follows. Firstly, we present a lazy learning method to minimize basically required learning time. In deep Q-network, there is a memory module for storing experience transactions. There is no need to train the neural network every time when we have to store a new transition. Because the memory space is very large and any small change in memory may not show any big difference. So we can apply lazy learning method to train the neural network more efficiently and stably. Secondly, we'd better remove the unnecessary repetition steps. At the beginning, the agent knows nothing about the environment and must have to choose any action randomly. Since these actions are randomly chosen, the agent always gets back to previous states and causes unnecessary repetitions, resulting in unnecessary execution time. Removing these highly correlated transitions, we can train the neural network more efficiently and stably. These two methods are applied jointly to enhance training performance without losing any accuracy.

We evaluate our method on the obstacle avoidance and path planning task, where an agent trapped in an unknown environment finds out the shortest path to the destination without any collision. And the experiment results show that the proposed method reduces 53.38% of training time on average compared to the traditional method with no performance loss and make the training procedure more stable. Thus, our method can be used effectively and stably in the applications of obstacle avoidance.

The rest of paper is organized as follows. In section 2, describes the related work. In section 3, presents the background. In section 3, introduces the proposed method. In section 4, experiments and results are provided. Finally, in section 5, gives the conclusions.

2. Related Work

In this section, some research related to deep Q-network will be introduced. Deep Q-network has been used for various purposes such as path planning [7], robotics [8], video games[9] and so on.

Tai Lei et al. [7] introduced a deep reinforcement learning method for achieving obstacle avoidance with the RGB-D sensor only. Depth images fed into the convolution neural network (CNN) and after that, feature map were extracted. Then through Q-learning network, the robot performed obstacle avoidance. The experiment results show that the deep Q-network is reliable to train a robot to achieve obstacle avoidance.

Shixiang Gu et al. [8] demonstrated that deep reinforcement learning can not only solve simple tasks but also can solve complex 3D manipulation tasks with only a few hours of training. In this paper, the author also presented a method that can reduce the training time by parallelizing the algorithm across two or more robots.

Volodymyr Mnih et al. [9] first presented the deep reinforcement learning model using raw pixels directly as input. The model is a convolution neural network with a Q-learning. They evaluate the method in seven Atari 2600 Arcade Learning Environment (ALE) and found that it outperformed on six of games compared with all previous approaches. Especially, it surpasses human experts on three of games.

However, most of the previous studies have failed to reduce the training time of deep reinforcement learning. Thus, they might spend a lot of time in training. In this paper, we consider the limitations of previous studies and provide a fast training method.

3. Effective Lazy Training Method

In this paper, we propose an effective lazy training method for deep reinforcement learning, especially for deep Q- network, combining a neural network with Q-learning scheme. Most of the previous studies have never reduced the training time of deep reinforcement learning. Thus, they might spend a lot of time in training. In this paper, the limitations of previous studies are analyzed to design a fast and stable training method. To achieve this goal, our specific method is described as follows. Firstly, we propose a lazy learning method. Secondly, we present a method of removing unnecessary repetition in training procedure.

3.1 Lazy Learning Method

In the deep Q-network, it selects samples through random mini-batch from the memory and trains the neural network using the gradient descent. We can easily find that, in the traditional deep Q-network algorithm [2] [9], it always tries to train after storing the transition. However, there is no need to train every time if the changes in memory are not big enough. Because the memory to store the transitions is very large and any small change in memory may not make any big difference. So we propose a lazy learning method to simplify the required training. Lazy learning method is that when any difference between evaluate network and target network becomes large, we need to train more times. On the contrary, when the difference between evaluate network and target network is small, we can train few times. The specific algorithm's pseudocode is shown as in below algorithm 1.

When storing a transition in memory *D*, we can call it as a step. As the algorithm shows, we calculate the difference between evaluate network and target network first and call it as a loss. If the loss is bigger than *T*, which is defined as a threshold, we need to train the neural network every C_1 steps. If the loss is smaller than *T*, we train the neural network every C_2 steps. And of course, C_1 is bigger than C_2. Training the neural network every *N* steps can also make the random mini-batch sample not correlated and train more efficiently and stability. And we can train the neural network after C_3 transitions stored in the memory *D*. Because at first, there is a little transition in the memory *D* and there is no need to train.

Algorithm 1: Lazy learning method

Initialize replay memory ***D***
Initialize ***step*** = 0;
Build target and evaluated Neural Network
For each episode
  Store the transitions (s_t, a_t, r_t, s_{t+1}) in memory ***D***
  step ++
  loss = Q_evaluate - Q_traget
  If (**loss** >***T***)
    N = C_1
  Else
    N = C_2
  If (***step*** > C_3) and (***step*** % ***N***== 0)
    Select a random mini-batch of transitions (s_k, a_k, r_k, s_{k+1}) from memory ***D***
    Update θ through a gradient descent on $(y_k\text{-}Q(s_k, a_k; \theta))^2$

3.2 Removing Unnecessary Repetition

At first, the agent knows nothing and must have to choose its required action randomly. Since these actions are randomly chosen, unnecessary repetitions may occur all the time. When the agent selects an action and moves to the next state, if it is the same as the previous state, it is called unnecessary repetition. For example, as shown in Figure 1, the red grid is the agent, the yellow grid is the goal, the gray grids are the walls and the black

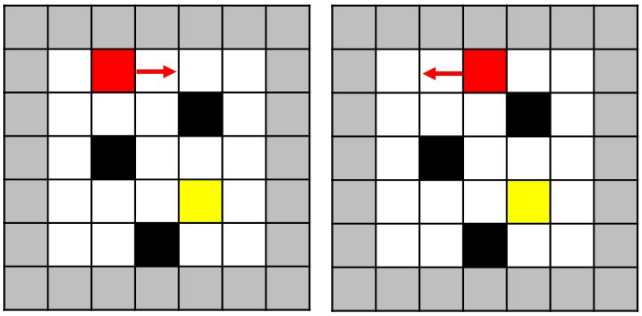


Fig.1. An example of removing unnecessary repetition.

grids are the obstacles. The previous state's action is determined as right action, as shown in the first figure. And next state's action is determined as left, as shown in the right figure. In this situation, the agent may return to the previous state which causes unnecessary repetition. Through removing unnecessary repetition, we can prevent the agent to store unnecessary correlated transitions and train more efficiently with the stability. The specific algorithm's pseudocode is described as below algorithm 2.

Algorithm 2: Removing unnecessary repetition

For each episode
  Initialize state s_t
  While state s_t is not terminal
    IF (probability < ε)
      Select $a_t = \max Q(s_t)$
    ELSE
      While (True)
        Select an action a_t randomly
        If (s_{t+1} == previous state)
          Continue
        Else
          Break
    Take action a_t, observe r_t and s_{t+1}

3.3 Total Proposed Method

We added lazy learning method and removing unnecessary repetitions to the traditional deep Q-network algorithm. The total improved deep Q-network algorithm's pseudocode is like below algorithm 3. And next section we will evaluate these proposed methods.

Algorithm 3: Improved Deep Q Network Algorithm

Initialize replay memory D
Initialize ***step*** = 0;
Build target and evaluated Neural Network
For each episode
  Initialize state s_t
  While state s_t is not terminal
    IF (probability < ε)
      select $a_t = \max Q(s_t)$
    ELSE
      While (True)
        Select an action a_t randomly
        If (s_{t+1} == previous state)
          Continue
        Else
          Break
    Take action a_t, observe r_t and s_{t+1}
    Store the transitions(s_t, a_t, r_t, s_{t+1}) in memory D
    step ++
    loss = Q_evaluate - Q_traget
    If (**loss** >T)
      $N = C_1$
    Else
      $N = C_2$
    If (***step*** > C_3) and (***step*** % N== 0)
      Select a random mini-batch of transitions (s_k, a_k, r_k, s_{k+1}) from memory D
      Set

$$y_k = \begin{cases} r_k & \text{for terminal } s_{k+1} \\ r_k + \gamma \cdot \max_{a'} Q(s_{k+1}, a'; \theta) & \text{for non-terminal } s_{k+1} \end{cases}$$

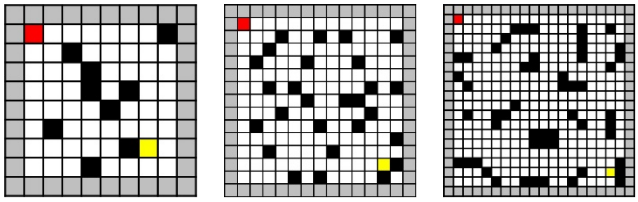
      Update θ through a gradient descent on $(y_k - Q(s_k, a_k; \theta))^2$
    Set the next state as the current state $s_t \leftarrow s_{t+1}$

4. Experiment and Results

We evaluate the proposed method on the obstacle avoidance problem and its associated path planning task, where an agent trapped in an unknown environment and through self-study will find out the shortest path to the destination without any collision.

4.1 Experimental Environment

We create a virtual space for the agent. Like previous research [10], we create different sizes of maze environment with randomly generated obstacles. As shown in Figure 2, we can create a 10×10 size maze with 9 obstacles, a 15×15 size one with 27 obstacles, and a 20×20 size one with 52 obstacles. And the red grid is the agent, the yellow grid is the goal, the gray grids are the walls and the black grids are the obstacles. A grid-based model has been used in many path planning and obstacle avoidance methods. And the agent can select up, down, right, left 4 actions. Then, the agent can find the quick way from start point to the end point through deep Q-network. The structure of the neural network is presented in Figure 3. Input data are map state features and the output data are every action's Q-values. Input layer's size is configured as 2×10, hidden layer's size is 10×10, and output layer's size is 10×4. Through the neural network, we can directly generate the Q-values. To build the neural network more easily, we used Tensorflow [11], which is an open source software library developed by researchers and engineers working on the Google Brain Team. The experiments are executed on an Intel® Core™ i3-2100 CPU. Through training many times, we get the best configurations of the deep Q-network. The training parameters and their values are shown as in Table Ⅰ.



a) 10×10 b) 15×15 c) 20×20

Fig.2. Maze environment

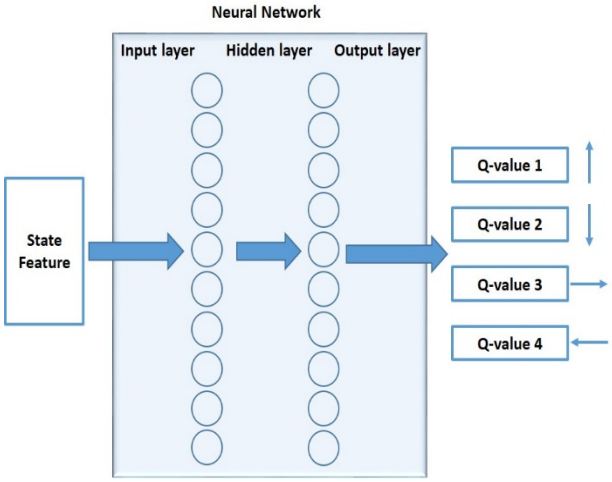


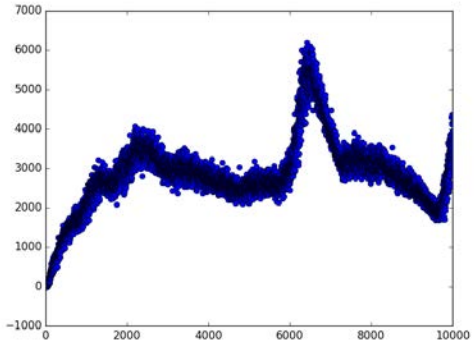
Fig.3. Neural network structure

TABLE I. TRAINING PARAMETERS AND VALUES

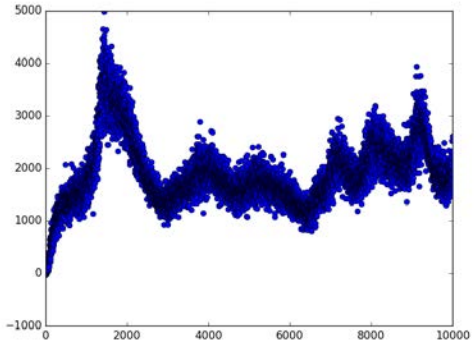
Parameter	Value
Learning rate	0.01
Discount factor	0.9
ε-greedy exploration	0.1
Mini-batch size	32
Reply memory size	5000
Reward of wall	-1
Reward of obstacle	-1
Reward of goal	5

4.2 Results of Lazy Learning

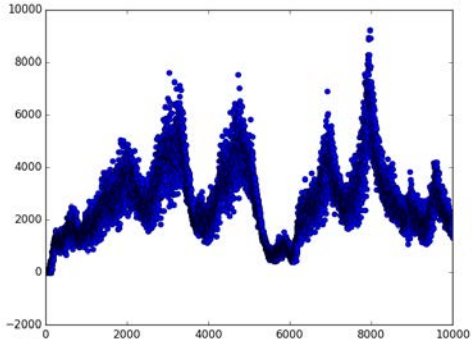
To evaluate how effective the method of lazy learning works, we do the experiment with different steps and maze environment first. The stability results about the traditional method are shown in Figure 4. The horizontal axis is the number of the episode and the vertical axis is the sum of the Q-value of four actions. We can find that when the sum of the Q-value is around 2500. So we can set 2500 as threshold. And the experiment results about training time with different steps and maze environment are shown in Figure 5. The horizontal axis is the step and the vertical axis is training time. Because the experiment is randomly trained, we trained every case 10 times and get their average training time. Through the figure, we can see that as the of steps grows, the training time can be reduced. When the step is 2, training time is reduced most. And when the step is bigger than 6, the training time is nearly not reduced, becoming stable. According to the result, when the loss is bigger than the 2500, we can set the step as 2 and when the loss is smaller than the 2500, we can set the step as 6. The specific result of the comparison between traditional method and lazy learning method are shown in Figure 6. In 10×10 environment, the training time is reduced from 464.639 to 242.567 and about 47.79%. In 15×15 environment, the training time is reduced from 828.339 to 435.007 and about 47.48%. In 20×20 environment, the training time is reduced from 1295.928 to 704.101 and about 45.46%. And average training time is reduced by about 46.91%.



a) 10×10 traditional method



b) 15×15 traditional method



c) 20×20 traditional method

Fig.4. Training stability of traditional method

4.3 Results of Removing Unnecessary Repetition

Before evaluating how well the method of removing unnecessary repetition method works, we measure the percentage of unnecessary repetitions and the results are shown in Table Ⅱ. In deep Q-network, with 0.1 probability, the agent will explore the new environment and select action randomly. And we find that, when selecting actions randomly, approximately 31.42% of these actions are unnecessarily repeated. So we can remove these unnecessary repetitions to

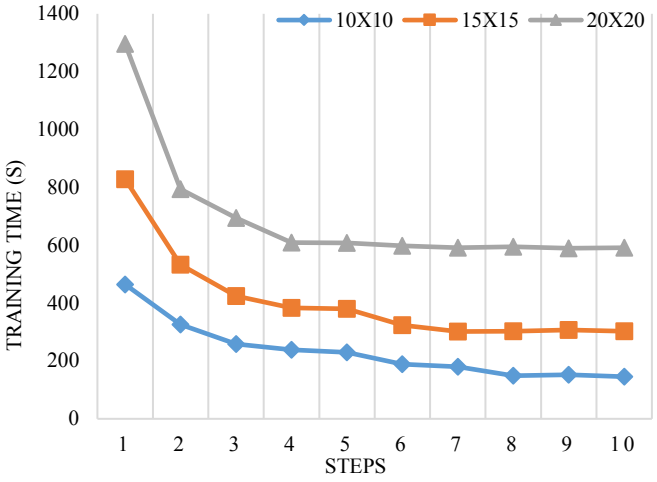


Fig.5. Training time with different steps

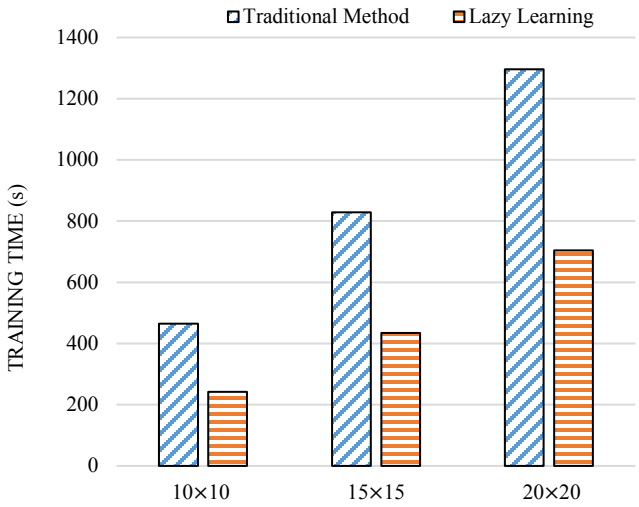


Fig.6. Comparison between traditional method and lazy learning method

reduce correlated transitions. The experiment results about removing unnecessary repetition method are shown in Figure 7. As we can see, the proposed method can also reduce the training time and train procedure more efficiently and stably than the traditional method. In 10×10 environment, the training time is reduced from 464.639 to 418.984 and about 9.83%. In 15×15 environment, the training time is reduced from 828.339 to 767.709 by about 7.32%. In 20×20 environment, the training time is reduced from 1295.928 to 1197.485 by about 7.24%. And average training time is reduced about 8.13%.

TABLE II. PERCENTAGE OF UNNECESSARY REPETITION

Maze Environment	Random Action	Unnecessary Repetition	Percentage (%)
10×10	13258	4027	30.37%
15×15	24435	7763	31.77%
20×20	36850	11839	32.13%
Average			31.42%

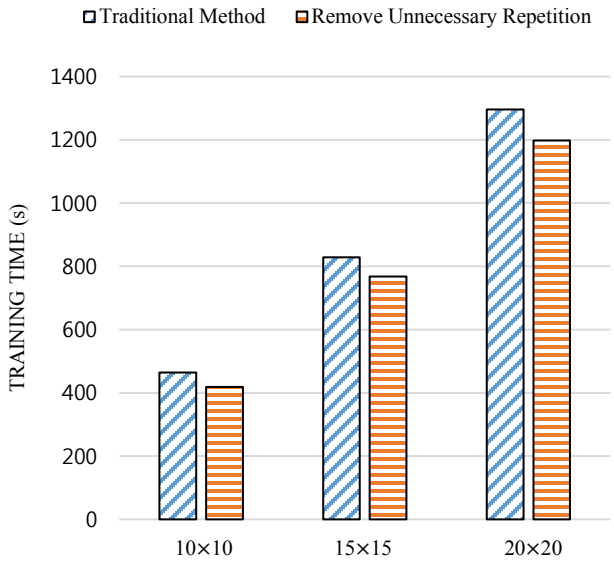


Fig.7. Comparison between traditional method and lazy learning method

4.4 Total Traning Time Comparison

In overall, we combine all proposed methods together and compare with the traditional method. The experiment results are shown in Figure 8. As we can see, the proposed method can reduce the training time significantly. In 10×10 environment, compared to the traditional method, the training time is reduced by about 55.77%. In 15×15 environment, compared to the traditional method, the training time is reduced by about 52.05%. In 20×20 environment, compared to the traditional method, the training time is reduced by about 52.32%. All in all, average training time can be reduced by about 53.38%.

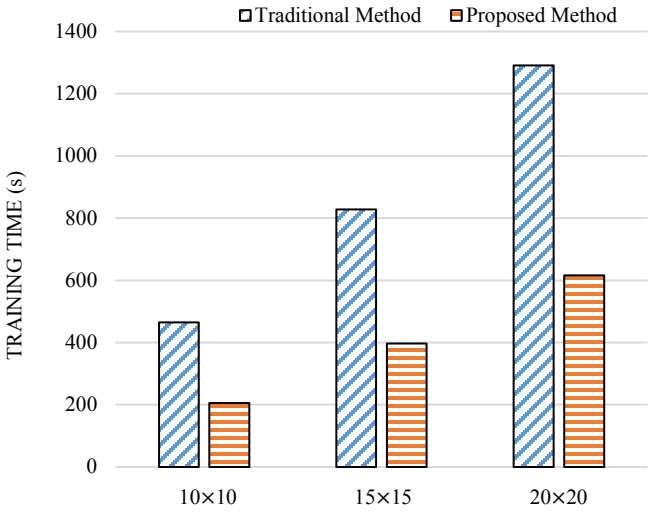
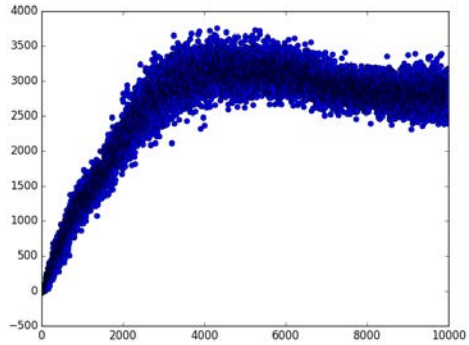


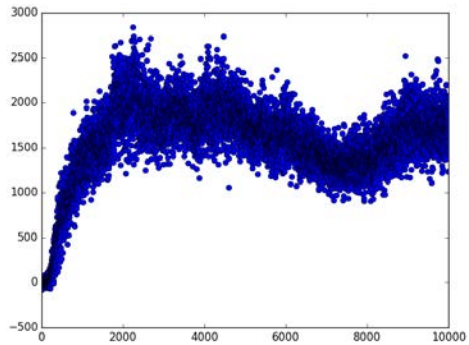
Fig.8. Total training time comparison

4.5 Training Stability Comparison

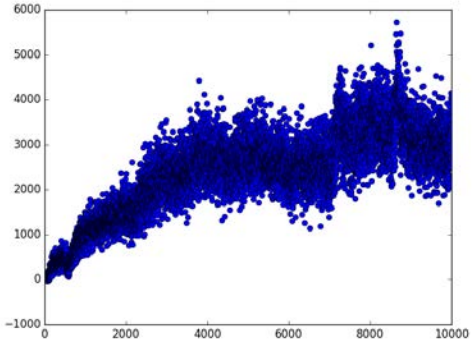
Besides evaluating the training time, we also compare the training stability between traditional method and the proposed method. And Figure 9 shows the training stability of traditional method. The horizontal axis is the number of the episode and the vertical axis is the sum of the Q-value of four actions. Through the figure, we can see that as the maze environment grows, the training procedure is more complicated. And we can also easily find that the proposed method not only improve training efficiency but also improve training stability.



a) 10×10 proposed method



b) 15×15 proposed method



c) 20×20 proposed method

Fig.9. Training stability of proposed method

5. CONCLUSION

In this paper, we have proposed an effective lazy training method for deep reinforcement learning, especially for deep Q-network. We evaluate our method on the obstacle avoidance and path planning task. Firstly, we proposed lazy learning method and the experiment results show that the training time is reduced 46.91% on average. Secondly, we remove the unnecessary repetitions and the experiment results show that the training time is reduced 8.13% on average. And also our combined effect as overall achievement shows that the proposed method reduces 53.38% of training time on average, compared to the traditional method with no performance loss. Besides reducing training time, the proposed method also can improve the stability of training. Through the experiment, we prove that the proposed method can significantly reduce the training time and make the training procedure more stable.

For the future work, we will try different methods to make the training procedure more efficient and stable. On the other hand, we will also try to evaluate our methods in various deep reinforcement learning environment.

ACKNOWLEDGMENT

This work was partially supported by Institute for Information & Communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (R0124-16-0002, Emotional Intelligence Technology to Infer Human Emotion and Carry on Dialogue Accordingly) and the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (NRF-2015R1A2A2A01007668).

REFERENCES

[1] R. S. Sutton, and A. G. Barto. Reinforcement learning: An introduction, vol. 1, MIT press Cambridge, 1998.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, et al, "Human-level control through deep reinforcement learning," Nature, vol.518(7540), pp.529-533, 2015.

[3] T. Zhang, Q. Li, C.-s. Zhang, H.-w. Liang, P. Li, T.-m. Wang, et al, "Current trends in the development of intelligent unmanned autonomous systems," Front. Inform. Technol. Electron. Eng, vol.18(1), pp.68-85, January 2017.

[4] W. Xia, H. Li, and B. Li, "A Control Strategy of Autonomous Vehicles Based on Deep Reinforcement Learning," 9th International Symposium on Computational Intelligence and Design, 2016, pp.198–201.

[5] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh, "Action-conditional video prediction using deep networks in atari games," Advances in Neural Information Processing Systems, 2015, pp.2863–2871.

[6] A. H. Qureshi, Y. Nakamura, Y. Yoshikawa, and H. Ishiguro, "Robot gains social intelligence through multimodal deep reinforcement learning," IEEE-RAS 16th International Conference on Humanoid Robots, 2016, pp.745–751.

[7] T. Lei, and L. Ming, "A robot exploration strategy based on q-learning network," IEEE International Conference on Real-time Computing and Robotics, 2016, pp.57–62.

[8] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-Policy Updates," arXiv preprint arXiv:1610.00633, November 2016.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, et al, "Playing atari with deep reinforcement learning," arXiv preprint arXiv:1312.5602, December 2013.

[10] J. Kim, S. Shin, J. Wu, S. D. Kim, and C. G. Kim, "Obstacle Avoidance Path Planning for UAV Using Reinforcement Learning Under Simulated Environment," IASER 3rd International Conference on Electronics, Electrical Engineering, Computer Science, Okinawa, 2017, pp.34–36.

[11] TensorFlow: https://www.tensorflow.org