International Journal of Internet, Broadcasting and Communication Vol.15 No.3 166-175 (2023)
http://dx.doi.org/10.7236/IJIBC.2023.15.3.166

IJIBC 23-3-21

Comparison of value-based Reinforcement Learning Algorithms in Cart-Pole Environment

Byeong-Chan Han*, Ho-Chan Kim**, Min-Jae Kang***

**Graduate student, Dept. of Electronic Engineering, Jeju National University, Korea*
gksqudcks@jejunu.ac.kr

***Professor, Dept. of Electrical Engineering, Jeju National University, Korea*
hckim@jejunu.ac.kr

****Professor, Dept of Electronic Engineering, Jeju National University, Korea*
minjk@jejunu.ac.kr

Abstract

Reinforcement learning can be applied to a wide variety of problems. However, the fundamental limitation of reinforcement learning is that it is difficult to derive an answer within a given time because the problems in the real world are too complex. Then, with the development of neural network technology, research on deep reinforcement learning that combines deep learning with reinforcement learning is receiving lots of attention. In this paper, two types of neural networks are combined with reinforcement learning and their characteristics were compared and analyzed with existing value-based reinforcement learning algorithms. Two types of neural networks are FNN and CNN, and existing reinforcement learning algorithms are SARSA and Q-learning.

Keywords: *Reinforcement learning, deep reinforcement learning, FNN, CNN, SARSA, Q-learning.*

1. INTRODUCTION

Reinforcement learning is a type of machine learning, and it is a field that allows self-learning through repetition by judging later whether an action is good or bad when an action is performed in a certain environment and giving a reward [1]. Machine learning is a study that studies the methodology in which machines or computers computationally simulate the process of human learning by imitating human learning and expressing it statistically or mathematically. Machine learning includes supervised learning, unsupervised learning, and reinforcement learning. Supervised learning is a method that reduces the error between the output of the input data and the label data as learning progresses because there is label data for all input data. Unsupervised learning is distinguished from supervised learning as it learns without label data, and in this method, data with similar characteristics are grouped together as learning progresses. That is, it aims to find the hidden structure of unlabeled data. On the other hand, reinforcement learning is different from other machine learning in that it learns through rewards obtained through trials and errors in given environment. As shown in Figure 1, reinforcement learning has two components: an agent that performs learning and an environment. The agent observes a specific environment and learns which action brings the maximum reward

Manuscript Received: July. 2, 2023 / Revised: July. 8, 2023 / Accepted: July. 11, 2023
Corresponding Author: minjk@jejunu.ac.kr
Tel: +82-64-754-3666, Fax: +82-64-756-1745
Professor, Dept. of Electronic Engineering, Jeju National Univ, Korea

among selectable actions. In this way, reinforcement learning learns when an agent receives rewards from a changing environment, which is similar to how a person learns through trial and error in the process of acquiring knowledge. And you need to be able to choose the action that brings you the maximum reward and strike the right balance between exploitation and exploration. Here, the exploitation is to perform an action that can obtain the maximum reward in the current state, and new attempts are needed to accumulate various experiences, and these new attempts are called exploration. The key principle of reinforcement learning is to have an appropriate balance between exploitation and exploration.

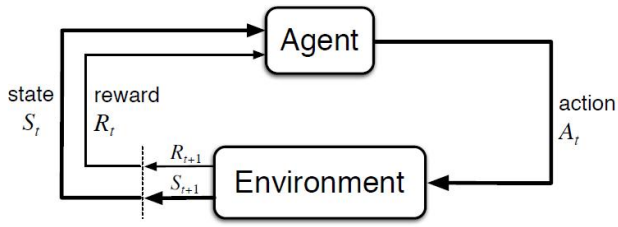


Figure 1. The agent-environment interaction in reinforcement learning [1].

Reinforcement learning can be applied not only to various types of games, but also to a wide range of problems in the real world, such as robot control, stock trading, resource allocation, recommendation systems, and natural language processing. In other words, while the application fields to which reinforcement learning can be applied are infinite, researchers have had difficulty solving real-world problems using reinforcement learning for a while. For problems with simple state space and action space, existing reinforcement learning algorithms could achieve good results, but the fundamental limitation is that it is impossible to derive an answer within a given time because problems in the real world are so complex. Then, deep neural network learning technology and convolutional neural network technology for image recognition were developed, and research on deep reinforcement learning technology combining deep learning with reinforcement learning was conducted, and the developed method is DQN (Deep Q-Network). Currently, many studies that apply deep reinforcement learning techniques to solve a wide range of problems in the real world are attracting attention.

The Cart-Pole balance problem is a standard problem widely used to evaluate the performance of reinforcement learning algorithms. In general, the performance evaluation of reinforcement learning to control a cart-pole system is the number of times (episodes) required to maintain balance so that the cart does not deviate from the range of the track and the pole does not fall over during the desired step of the learning system. In addition, the stability of the learning system can be evaluated by measuring the change rate of the pole slope during the learning period. In this paper, the Cart-Pole system was used to analyze the characteristics of the existing reinforcement learning algorithm and the reinforcement learning algorithm using deep learning, and the Gym library provided by OpenAI was used as the experimental environment. Existing reinforcement learning algorithms used Q-learning and SARSA, and reinforcement learning algorithms using deep learning used FNN and CNN.

The rest of this paper is organized as follows. Chapter 2 explained the Cart-Pole system, and Chapter 3 explained the value-based reinforcement learning algorithm and the Q-learning, SARSA, and DQN algorithms that represent this learning method. And in Chapter 4, the convergence speed and agent stability were tested for performance analysis of these algorithms. In Chapter 5, conclusions and directions for future research are presented.

2. Cart-pole system

The Cart-Pole system is a standard problem widely used to evaluate the performance of reinforcement learning algorithms. The cart-pole control problem consists of a cart and poles counted perpendicular to the cart, as shown in Figure 2.

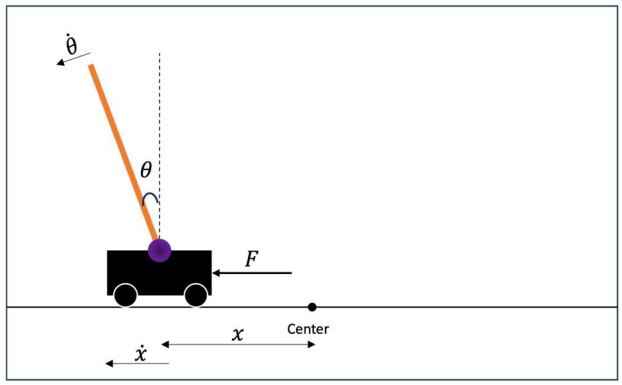


Figure 2. Cart-Pole environment

The cart can freely move up and down the horizon without friction. The agent can apply a constant force to the cart to the left or right. The goal is to keep the pole from tipping over and hold it for as long as possible. In the Cart-Pole learning system, a constant force is applied to the cart in each learning step, and four state values $[x, \dot{x}, \theta, \dot{\theta}]$ are observed as shown in Table 1.

Table 1. Observation space of Cart-Pole environment.

Observation Space			
	Observation	Min	Max
x	Cart Position	-4.8	$+4.8$
$\dot{x}$	Cart Velocity	$-\infty$	$+\infty$
θ	Pole Angle	-0.418	$+0.418$
$\dot{\theta}$	Pole Angular Velocity	$-\infty$	$+\infty$

The state of the system is defined by the angle θ and angular velocity $\dot{\theta}$ of the pole, and the linear position x and velocity $\dot{x}$ of the cart [2] as follows.

$$\ddot{\theta} = \frac{(M+m)g \sin\theta - \cos\theta[F+ml\theta^2 \sin\theta]}{\left(\frac{4}{3}\right)(M+m)l - ml\cos^2\theta} \quad (1)$$

$$\ddot{x} = \frac{F+ml[\theta^2 \sin\theta - \ddot{\theta}\cos\theta]}{(M+m)} \quad (2)$$

Where M is the mass of the cart $(1.0\ kg)$, is the mass of the pendulum $(0.1\ kg)$, is the gravitational acceleration $(9.8\ m/s^2)$, F is the force applied to the cart (± 10 Newtons) and l is the length from the center of gravity to the pendulum $(0.5\ m)$.

3. Value-Based Reinforcement Learning

There are two main approaches to solving reinforcement learning problems. That is, a method based on a value function and a method based on a policy. Of course, there is also a hybrid type approach that mixes value function and policy search. The value function method is based on an estimate of a given value (expected reward) in given state [3-4]. The state-value function is the expected return starting at state s. It is expressed as

$$V^{\pi}(s) = E(R|s, \pi) \quad (3)$$

The optimal policy, π^*, has a corresponding state-value function $V^*(s)$, and the optimal state-value function is defined as follows.

$$V^{\pi*}(s) = \max_{\pi} V^{\pi}(s) \quad \forall s \in S \tag{4}$$

That is, the optimal policy is to select the value with the largest expected return (rewards) among all possible actions (a) in the current state. Here, we define a new state-action value function $Q^{\pi}(s,a)$ that is similar to the state value function but adds action (a) to the variable.

$$Q^{\pi}(s,a) = E(R|s,a,\pi) \tag{5}$$

The best policy is to choose the action that maximizes the state-action value function in given state. in other words,

$$Q^{\pi*}(s,a) = \max_{\pi} Q^{\pi}(s,a) \quad \forall s \in S, \forall a \in A \tag{6}$$

Reinforcement learning methods based on the above equation include Q-learning, SARSA, and DQN. The learning algorithms of Q-learning and SARSA are very similar. And DQN is a value-based reinforcement learning method that combines deep learning.

3.1. Traditional reinforcement learning

In the traditional reinforcement learning, after updating all values according to state-action in the form of a table (Q-Table), the policy was configured to select an action by referring to it. The policy is determined based on the current Q-Table, but if the updating policy and the target policy are the same, SARSA, which is on-policy, and Q-Learning, which is off-policy, if the two policies are different. Either way, traditional reinforcement learning is based on tables and must have all the values for state-action, so if there are a lot of states, you need to have a very large table.

3.1.1. Q-learning

Reinforcement learning based on Q-learning is a method of determining which action to take in which state after approximating the value for each state-action pair [5-7]. The update equation of the function is:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_t + \gamma \max_{a+1} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]. \tag{7}$$

Here, (s_t, a_t) corresponds to the current state and action, (s_{t+1}, a_{t+t}) corresponds to the next state and action, $Q(s_t, a_t)$ is the value of the function that the agent took action a_t in the current state s_t, and r_t is the reward obtained by taking action in the current state s_t. In Equation (7), α is the learning rate and $\gamma(0 < \gamma < 1)$ is the discount factor. Each stage of Q-learning updates the state-action value table. That is, the internal state s_t of the Cart-Pole dynamic expressed in a form interpretable by the agent in the current state s_t, and the action a_t to be performed in the current state. For the Cart-Pole problem, the internal states of the mechanics are the position of the cart and the tilt angle of the pole, and the action is to accelerate the cart uniformly to the left or right on the track. Q-learning is classified as an off-policy method because it uses two different policies: the one used to update a function and the one used to find the next action. The policy to update the function is $\max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$, a greedy policy given by Eq. (7). However, the search policy for the next action generally uses the ε-greedy method. That is, the selection of the next action uses a mixture of greedy policies while adjusting the random search rate with the size of ε. In a Cart-Pole environment, the reward value r_t, is given by the reinforcement learning agent as 0 if the cart or pole exceeds the limit defined in the goal, and +1 otherwise.

3.1.2. SARSA

SARSA stands for "State-Action-Reward-State-Action" and works by iteratively updating action values based on transitions observed in the environment. Q-learning and SARSA algorithms are very similar. The biggest difference is that Q-learning follows an off-policy and SARSA follows an on-policy [8-10]. That is, in SARSA, the selection policy of the next state action is the same as the policy used when updating the function. As shown in Figure 3, SARSA selects the next state action in an ε-greedy way and updates the Q table with the selected action. On the other hand, Q-learning selects the next state action in the same way but chooses a policy that maximizes the reward when updating the Q table.

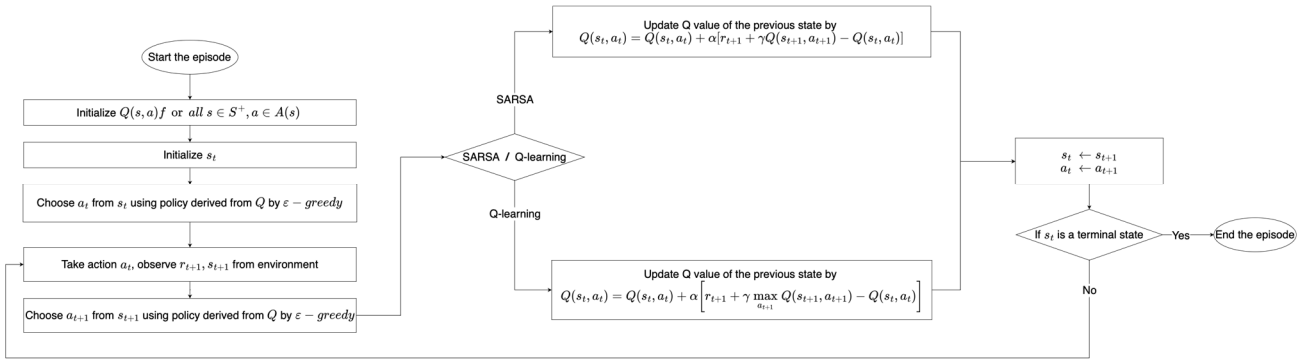


Figure 3. Flowchart of SARSA and Q-learning.

3.2. Reinforcement learning with Neural Network

DQN is an algorithm created by combining Q-learning and deep learning. Existing Q-learning learns by storing values corresponding to possible states and actions in a table format. This method requires a lot of memory and search time because the table grows as the state space and action space grow. To solve this problem, DQN combined with deep learning has been proposed. If the function corresponding to the table can be approximated with a nonlinear function using deep learning, there is no need to find or store values. However, if you simply try to apply deep learning, the following problems arise. Deep learning trains under the assumption that data with labels (correct answers) and data are independent of each other [11]. On the other hand, in reinforcement learning, this assumption does not hold because the data do not have labels and the next state of the data is highly correlated with the current state. The proposed method to solve the problem of independence of data is a method called Experience Replay. In simple terms, it is a method of learning by collecting [current state, action, reward, next state] datasets for each time-step while proceeding with reinforcement learning episodes. If the data collected in this way is selected randomly, the correlation between each data is reduced, so the problem of data independence can be solved to some extent. And the problem of label data is solved by creating an additional Target network. That is, in the response memory, the current state data is randomly assigned to Q-network, the next state data is input to Target-network, and the output of Target-network is replaced by label data. The loss function of DQN is as follows.

$$L_j(\theta) = (y_i - Q(s_j, a_j; \theta))^2 \tag{8}$$

Here, $Q(s_j, a_j; \theta)$ is calculated using the main Q-network as an action-value, and y_j is calculated using the target network as a target value as follows.

$$y_j = r_j + \gamma \max_{a_{j+1}} \widehat{Q}(s_{j+1}, a_{j+1}; \theta^-) \tag{9}$$

The algorithm of DQN updates the main Q-network θ at every step so that the loss function equation (8) is

minimized. And the target-network θ^- is updated to the main Q-network at regular step intervals. Figure 4. shows the schematic diagram of the DQN algorithm.

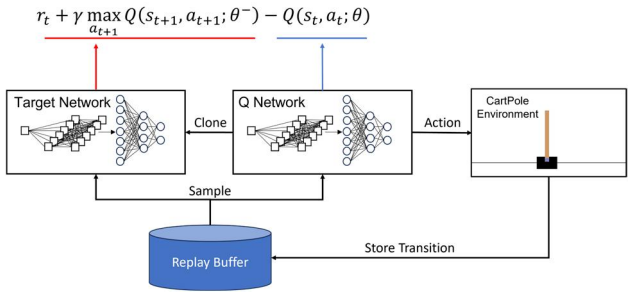


Figure 4. Schematic diagram of DQN algorithm.

4. EMPIRICAL RESULTS AND OBSERVATION

Representative algorithms Q-learning, SARSA, and DQN were selected for comparison of the value-based reinforcement learning algorithms of the Cart-Pole system. We used Cart-Pole environment from the Gym library provided by OpenAI, and the algorithm for the experiment was implemented with NumPy and Pytorch libraries.

To evaluate the performance of the learning speed and learning stability of the algorithm, the learning time required to reach the desired reward value and the change in pole slope were measured. Hyper parameters for Q-learning and SARSA algorithms used the same values as shown in Table 2.

Table 2. Hyper parameters of SARSA algorithm.

Hyper Parameters	
γ	0.95
α	0.25
ε	0.2

The hyper parameter values for the DQN algorithm are presented in Table 3. The parameter ε for the search policy of the next action was initially set to 0.9 to increase the random ratio to focus on search, and as learning progressed, it was set to 0.01 after 5000 steps to put weight on the greedy policy [12]. And the target update interval to update target-network θ^- to main Q-network θ was 50 steps. In the DQN algorithm, the neural network for deep learning uses two types of neural networks, namely FNN (Linear layer) and CNN, and compares the results. Both neural networks used the RMSprop optimizer, and the loss function used the Huber loss function. The input of the FNN neural network is received as a state vector provided by the environment, such as Q-learning and SARSA, and the input of the CNN is the Cart-Pole environment itself, not the Cart-Pole state vector, as a 60x90 pixel RGB image, and the convolution filter is set to 5x5 and the stride to 2.

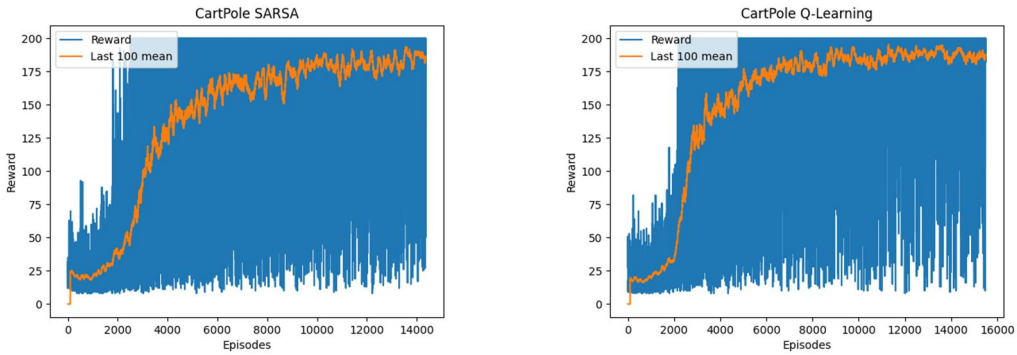
Table 3. Hyper parameters of DQN algorithm.

Hyper Parameters	
Batch Size	128
Batch of Frames	2

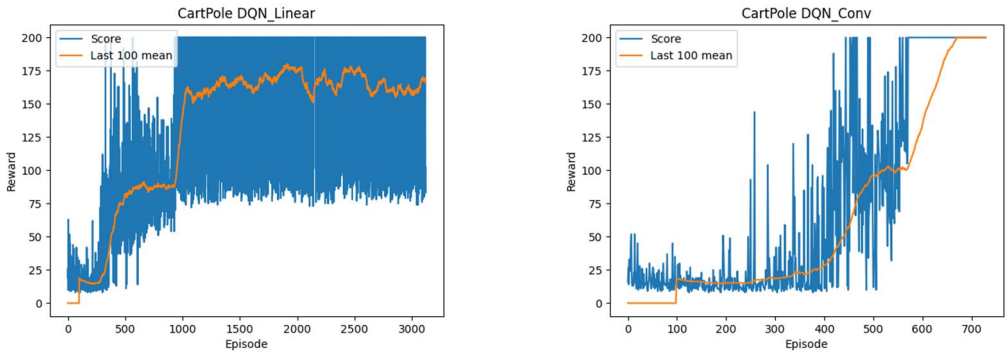
γ	0.999
init ε	0.9
final ε	0.01
ε Decay	5000
Memory Buffer Size N	100000
Target Update Frequency C	50

4.1. RL Reward of cart-pole system

In Figure 5, the blue line represents the reward value of each episode. The reward value of each episode has a large fluctuation amplitude, so it is difficult to determine the learning progress rate. Therefore, the average reward value of 100 episodes was displayed as an orange line to make it easier to recognize the learning progress. As shown in Figure 5, when the Q-learning algorithm is used, the reward reaches 200 for the first time from approximately the 2000th episode, similar to the SARSA algorithm. The average reward value converges faster in the Q-learning algorithm than when SARSA is used, but the oscillation width of the reward value is still large. DQN achieves a reward value of 200 much faster than SARSA and Q-learning algorithms for both algorithms using FNN and CNN, and the amplitude of oscillations is small. When using the DQN algorithm using CNN, the reward value reaches 200 from the 600th episode, which is much faster than SARSA and Q-learning as well as the DQN algorithm using FNN, and the average reward value completely reaches 200 from the 650th episode. From this, it can be seen that learning using a larger state space through RGB images is more helpful than learning using a state vector provided by the environment. This will become even more significant as the environment becomes more complex.



(a) SARSA (b) Q-learning

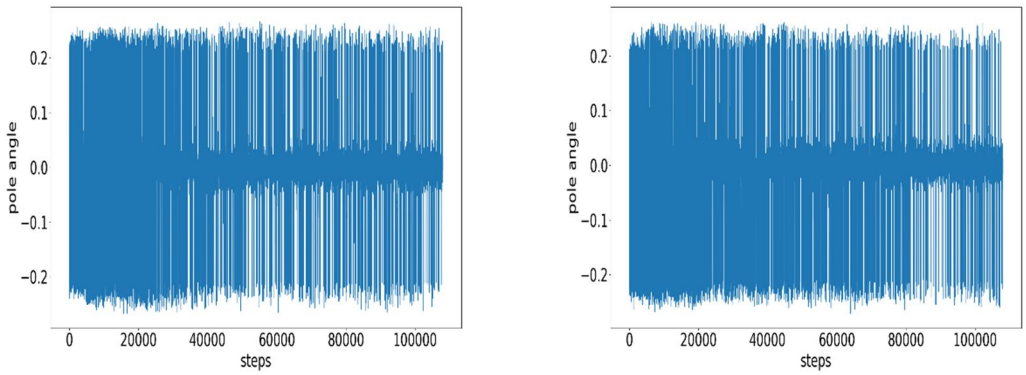


(c) DQN Linear (d) DQN Convolution

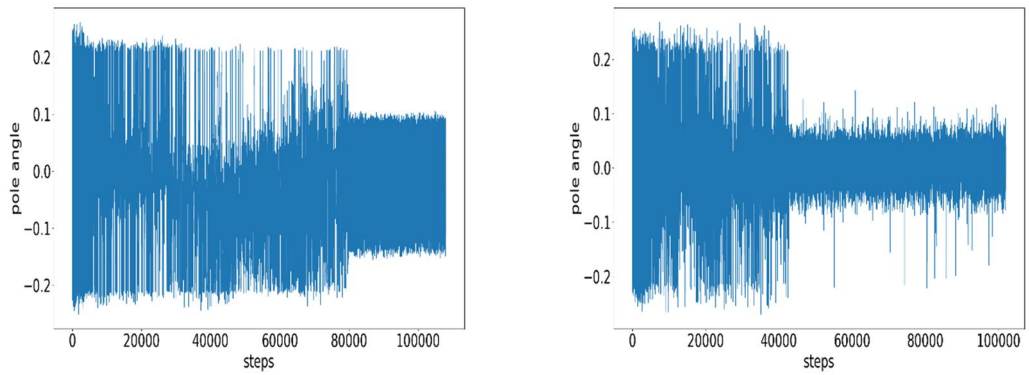
Figure 5. RL rewards of Cart-Pole system.

4.2. Pole Oscillation Angle of cart-pole system

The purpose of cart-pole control is to keep the pole from falling over for a long time, so an algorithm that keeps the pole angle constant is a good algorithm. In the case of learning using SARSA and Q-learning, there is a slight convergence after about 30,000 steps, but it does not converge completely and oscillates. Looking at the two DQN algorithms, when FNN is used, it converges more than when SARSA and Q-learning algorithms are used after about 80,000 steps, but the oscillation width of the value is slightly wider. In the case of using CNN, after 40000 steps, which is twice as fast as when using FNN, the angle of the pole converges to close to 0 radian, and it can be seen that the oscillation width is much smaller. Therefore, the DQN algorithm using CNN has an excellent ability to keep the angle of the pole constant, so it can be seen as a more effective algorithm in the Cart-Pole control environment than the rest of the algorithms.



(a) SARSA (b) Q-learning



(c) DQN Linear (d) DQN Convolution

Figure 6. Pole oscillation angle of Cart-Pole system using RL.

5. CONCLUSION

In this paper, the Cart-Pole system was used to analyze the characteristics of reinforcement learning that combines existing reinforcement learning and neural networks. The convergence speed of each algorithm required to learn the Cart-Pole as a system capable of maintaining balance for 200 steps and the change in the slope of the pole were measured. The convergence speed was similar between Q-learning and SARSA, and DQN was faster than these algorithms. In particular, DQN using CNN showed no oscillation after reaching 200 steps, showing superiority over other algorithms. In addition, to analyze the stability of the system, the vibration width of the pole was observed, and the vibration amplitude of the pole of DQN using CNN converged to the smallest, and the convergence speed was more than twice as fast as other algorithms. As a future research project, we plan to conduct research on reinforcement algorithms that combine different types of deep learning neural networks.

ACKNOWLEDGEMENT

This research was supported by the 2023 scientific promotion program funded by Jeju National University

REFERENCES

[1] Richard Sutton and Andrew Barto. Reinforcement Learning: An Introduction. MIT Press, 1998.
[2] BROCKMAN, Greg, et al. Openai gym. arXiv preprint arXiv:1606.01540, 2016.
[3] D. -H. Lee, V. V. Quang, S. Jo and J. -J. Lee, "Online Support Vector Regression based value function approximation for Reinforcement Learning," *2009 IEEE International Symposium on Industrial Electronics*, Seoul, Korea (South), 2009, pp. 449-454, doi: https://doi.org/10.1109/ISIE.2009.5222726.
[4] Rammohan, Sreehari, et al. "Value-Based Reinforcement Learning for Continuous Control Robotic Manipulation in Multi-Task Sparse Reward Settings." *arXiv preprint arXiv:2107.13356*(2021).
[5] WATKINS, Christopher JCH; DAYAN, Peter. Q-learning. Machine learning, 1992, 8: 279-292.
[6] D. Pandey and P. Pandey, "Approximate Q-Learning: An Introduction," *2010 Second International Conference on Machine Learning and Computing*, Bangalore, India, 2010, pp. 317-320, doi: https://doi.org/10.1109/ICMLC.2010.38.
[7] N. Kantasewi, S. Marukatat, S. Thainimit and O. Manabu, "Multi Q-Table Q-Learning," *2019 10th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES)*, Bangkok, Thailand, 2019, pp. 1-7, doi: https://doi.org/10.1109/ICTEmSys.2019.8695963.
[8] RUMMERY, Gavin A.; NIRANJAN, Mahesan. On-line Q-learning using connectionist systems. Cambridge, UK: University of Cambridge, Department of Engineering, 1994.

[9] L. Harwin and S. P., "Comparison of SARSA algorithm and Temporal Difference Learning Algorithm for Robotic Path Planning for Static Obstacles," *2019 Third International Conference on Inventive Systems and Control (ICISC)*, Coimbatore, India, 2019, pp. 472-476, doi: https://doi.org/10.1109/ICISC44355.2019.9036354.

[10] T. Lu, K. Zhang and Y. Shi, "Sarsa-based Model Predictive Control with Improved Performance and Computational Complexity," *2022 IEEE 5th International Conference on Industrial Cyber-Physical Systems (ICPS)*, Coventry, United Kingdom, 2022, pp. 01-06, doi: https://doi.org/10.1109/ICPS51978.2022.9816896.

[11] MNIH, Volodymyr, et al. Human-level control through deep reinforcement learning. nature, 2015, 518.7540: 529-533.

[12] L. Hou, Z. Wang and H. Long, "An Improvement for Value-Based Reinforcement Learning Method Through Increasing Discount Factor Substitution," *2021 IEEE 24th International Conference on Computational Science and Engineering (CSE)*, Shenyang, China, 2021, pp. 94-100, doi: https://doi.org/10.1109/CSE53436.2021.00023.