Deep Reinforcement Learning with Experience Replay Based on SARSA

Dongbin Zhao, Haitao Wang, Kun Shao and Yuanheng Zhu
Key Laboratory of Management and Control for Complex Systems
Institute of Automation Chinese Academy of Sciences, Beijing 100190, China
dongbin.zhao@ia.ac.cn, wanghaitao8118@163.com, shaokun2014@ia.ac.cn, yuanheng.zhu@ia.ac.cn

***Abstract*—SARSA, as one kind of on-policy reinforcement learning methods, is integrated with deep learning to solve the video games control problems in this paper. We use deep convolutional neural network to estimate the state-action value, and SARSA learning to update it. Besides, experience replay is introduced to make the training process suitable to scalable machine learning problems. In this way, a new deep reinforcement learning method, called deep SARSA is proposed to solve complicated control problems such as imitating human to play video games. From the experiments results, we can conclude that the deep SARSA learning shows better performances in some aspects than deep Q learning.**

Keywords—SARSA learning; Q learning; experience replay; deep reinforcement learning; deep learning

I. INTRODUCTION

With the development of artificial intelligence (AI), more and more intelligent devices come into use in our daily lives. In the face of unknown complicated environment, these intelligent devices should know how to perceive the environment and make decisions accordingly.

In the last 20 years, deep learning (DL) [1] based on neural networks has greatly promoted the development of high dimension information perception problems. With the powerful generalizing ability, it can retrieve highly abstract structure or feature from the real environment and then precisely depict the complicated dependencies between raw data such as image and video. With excellent ability of feature detection, DL has been applied to many learning tasks, such as handwritten digit recognition [2], scenario analysis [3] and so on. Although it has achieved great breakthrough in information perception especially in image classifying problem, DL has its natural drawbacks. It can not directly select policy or deal with decision-making problems, resulting in limited applications in intelligent control field.

Different from DL, reinforcement learning (RL) is one class of methods which try to find optimal or near-optimal policy for complicated systems or agents [4-6]. As an effective decision-making method, it has been introduced into optimal control [7], model free control [8, 9] and so on. Generally, RL has two classes – policy iteration and value iteration. On the other hand, it can also be divided into off-policy and on-policy method. Common RL methods include Q learning, SARSA learning, TD(λ) and so on [4]. Though RL is naturally designed to deal with decision-making problems, it has run into great difficulties when handling high dimension data. With the development of feature detection method like DL, such problems are to be well solved.

A new method, called deep reinforcement learning (DRL), emerges to lead the direction of advanced AI research. DRL combines excellent perceiving ability of DL with decision-making ability of RL. In 2010, Lange [10] proposed a typical algorithm which applied a deep auto-encoder neural network (DANN) into a visual control task. Later, Abtahi and Fasel [11] employed a deep belief network (DBN) as the function approximation to improve the learning efficiency of traditional neural fitted-Q method. Then, Rrel [12] gave the complete definition of deep reinforcement learning in 2012. Most importantly, the group of DeepMind introduced deep Q network (DQN) [13, 14] which utilizes convolution neural network (CNN) instead of traditional Q network. Their method has been applied to video game platform called Arcade Learning Environment (ALE) [15] and can even obtain higher scores than human player in some games, like breakout. Based on their work, Levine [16] applied recurrent neural network to the framework proposed by DeepMind. In 2015, DeepMind puts forward a new framework of DRL based on Monte Carlo tree search (MCTS) to trained a Go agent called AlphaGo [17], which beats one of the most excellent human players Lee Sedol in 2016. This match raises the people's interest in DRL which is leading the trends of AI. Though DQN algorithm shows excellent performance in video games, it also has drawbacks, such as low efficient data sampling process and defects of off-policy RL methods.

In this paper, we focus on a brand new DRL based on SARSA learning, also called deep SARSA for imitating human players in playing video games. The deep SARSA method integrated with experience replay process is proposed. To the best of our knowledge, this is the first attempt to combine SARSA learning with DL for complicated systems.

The paper is organized as follows. In Section II, SARSA learning and ALE are introduced as the background and preliminary. Then a new deep SARSA is proposed in Section III to solve complicate control tasks such as video games. Two simulation results are given to validate the effectiveness of the proposed deep SARSA in Section IV. In the end we draw a conclusion.

This work was supported in part by National Natural Science Foundation of China (Nos. 61273136, 61573353, 61533017 and 61603382).

II. SARSA LEARNING AND ARCADE LEARNING ENVIRONMENT

A. Q learning and SARSA learning

Considering a Markov decision process (MDP), the goal of learning task is to maximize the future reward when the agent interacts with environment. Generally, we define the future reward from the time step t as

$$R(t) = \sum_{k=0}^{T} \gamma^k r_{t+k+1} \ , \tag{1}$$

where $\gamma \in (0,1]$ is the discount factor, r_t is the reward when an action is taken at time t and T is often regarded as the time when the process terminates. In addition, $T = \infty$ and $\gamma = 1$ can't be satisfied simultaneously. Then the state-action value function can be defined as $Q^{\pi}(s,a)$ which indicates the agent takes the action a at the state s under the policy π. Therefore,

$$\begin{aligned} Q^{\pi}(s,a) &= E_{\pi}\{R(t) \mid s_t = s, a_t = a\} \\ &= E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\} \end{aligned}, \tag{2}$$

where $E_{\pi}\{R(t) \mid s_t = s, a_t = a\}$ is the expected return, and π is the policy function over actions. Now the learning task aims at obtaining the optimal state-action function $Q^*(s,a)$ which is usually relevant to Bellman equation. Then two methods called Q learning and SARSA learning will be compared to get the optimal state-action value function.

As one of the traditional RL algorithms, Q learning is an off-policy method. The agent independently interacts with the environment which often indicates selecting the action a. Then the reward r is feedback from the environment and the next state s' is derived. Here $Q(s,a)$ represents the current state-action value. In order to update the current state-action value function, we employ the next state-action action value to estimate it. Although the next state s' has been given, the next action a' is still unknown. So the most important principle in Q learning is to take a greedy action to maximize the next $Q(s',a')$. The update equation is

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)] \ , \tag{3}$$

where α represents the learning rate.

By contrast, SARSA learning is an on-policy method. It means when updating the current state-action value, the next action a' will be taken. But in Q learning, the action a' is completely greedy. Given such analysis, the update equation of state-action value can be defined as

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma Q(s',a') - Q(s,a)\right] \ . \tag{4}$$

Actually, the difference between SARSA learning and Q learning lies in the update equations (3) and (4). In SARSA learning, the training data is quintuple-(s,a,r,s',a'). In every update process, this quintuple will be derived in sequence. However, in Q learning a' is just for estimation and will not be taken in fact.

B. Arcade Learning Environment

Arcade Learning Environment (ALE) is a wrapper or platform including many video games for Atari 2600. As a benchmark for new advanced RL algorithms, it presents some interferences for the agents, including the states and the rewards [15]. The states are high dimensional visual input ($210 \times 160 RGB$ video at 60 Hz) as what human receives and the reward can be transferred from the scores given by the environment when the agent interacts with the platform. Usually, the agent interacts with ALE through a set of 18 actions, but only 5 of which are basic actions. These 5 actions contain 4 movement directions, and the remaining one should be firing or null. To be clear, the reward or score comes from the system output instead of recognizing from the image.

ALE is designed to be remarkably suitable for testing RL algorithms. So many groups have applied their algorithms to this platform including the DQN method proposed by DeepMind. Though DQN has achieved excellent performance in video games, it only combines basic Q learning with deep learning. Many other different reinforcement learning methods can help improve the performance of deep reinforcement learning like the on-policy methods. In the next Section, we will present a DRL method based on SARSA learning to improve the training process in video games from ALE.

III. DEEP REINFORCEMENT LEARNING METHOD BASED ON SARSA LEARNING

Before deep reinforcement learning algorithms come out, many traditional RL methods have been applied to ALE. Deafzio and Graepel [18] applied some RL methods to those complicated video games. They compared the advantages and disadvantages of different RL methods such as Q learning, SARSA learning, actor-critic, GQ, R learning and so on. The results are listed in Table 1.

TABLE I. THE PERFORMANCE OF DIFFERENT RL METHODS IN ALE [18]

	SARSA	***AC***	***GQ***	***Q***	***R***
Relevant performance	1.00	0.99	0.65	0.82	0.96

From Table 1, the average performance of Q learning is only 82% of SARSA learning in video games. Though these algorithms only use hand craft features, the results above indicate that SARSA learning will achieve better performance than Q learning. So given on these facts, a new deep reinforcement learning method based on SARSA learning is proposed as follows.

A. SARSA network

Games from Atari 2600 can be regarded as a MDP which will be solved by RL algorithms. Here SARSA learning will be integrated to DRL framework. Similar to DQN in [14], given the current state s, the action a is selected by ε-greedy method. Then the next state s' and the reward r will be observed. The current state-action value is $Q(s,a)$. So in DRL based on SARSA, the current optimal state-action can be estimated by

$$Q^*(s,a) = E\left[r + \gamma Q(s',a') \mid s,a\right], \tag{5}$$

where a' is the next action selected by ε-greedy. Similarly, in deep SARSA learning, the value function approximation is still with the convolution neural network (CNN) whose structure is shown in Fig. 1.

The input of the CNN is the raw images from video games and the output is the Q values of all actions. θ is defined as parameters of the CNN. At the i_{th} iteration of training, the loss function of the network can be defined as

$$L_i(\theta_i) = (y_i - Q(s,a;\theta_i))^2, \tag{6}$$

where $y_i = r + \gamma Q(s',a';\theta_{i-1})$. Then the main objective is to optimize the loss function $L_i(\theta_i)$. From the view of supervised learning, y_i is regarded as the label in training though y_i is also a variable. By differentiating (6), we get the gradient of the loss function

$$\nabla L_i(\theta_i) = (r + \gamma Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i))\nabla Q(s,a;\theta_i), \tag{7}$$

where $\nabla Q(s,a;\theta_i)$ is the gradient of the current state-action value. Then according to (7), we can optimize the loss function by stochastic gradient descent (SGD), Adadelta and so on [19]. Besides, the reinforcement learning process should also be taken into consideration. The last layer of the network outputs the Q value of each action. So we can select the action and update it by the SARSA method. Fig. 2 depicts the forward data flow of SARSA network in the training process.

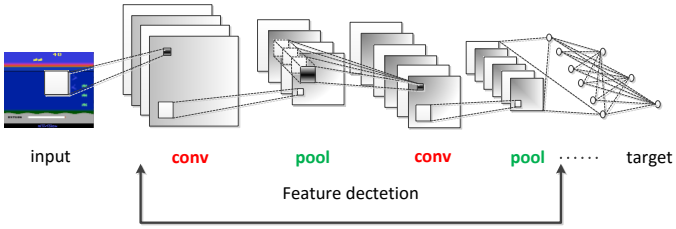

Fig. 1 The convolution neural network in DRL

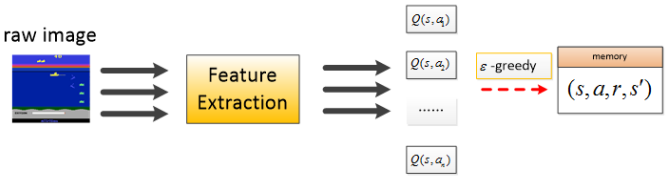

Fig. 2 The forward data flow in DRL

The "Feature Extraction" in Fig. 2 can be seen as the image preprocess and CNN network. After state-action is obtained, proper action is selected to make decision by the SARSA learning method. Later we introduce "experience replay" technique [20] to improve the training process of DRL and adapt reinforcement learning to scalable machine learning process.

B. Experience replay

In traditional reinforcement learning method, the learning and updating should continue in sequence. That is to say, every sample can stimulate one update, thus making the learning process rather slow. In order to adapt to the scalable machine learning process, the historical data is stored in memory and will be retrained later continuously. In Fig. 2, a quadruple $e_t = (s,a,r,s')$ is kept in historical data $D = e_1, \cdots$, where N indicates the size of historical stack. Then in the training process, we sample the training data from this stack D. There are some methods in which samples can be obtained, such as consecutive sampling, uniform sampling, and weighted sampling method by rewards. Here we follow the method of uniform sampling method in DQN, which has two advantages. Firstly, the efficiency of data usage is improved. Then consecutive samples might be greatly relevant to each other. Uniform method $(s,a,r,s') \sim U(D)$ can reduce the correlations between input data [14].

Before raw images from video games are sampled, some preprocess must be dealt with. We can obtain every frame from the video. However, it would be less efficient if one frame is regarded as the state s. Additionally, consecutive frames might contain important feature in images as the speed or geometrical relationship which can contribute more to the performance of agents. Once a single frame is trained, all those vital features are abandoned. So, in this paper one action is taken with every 4 frames, the same as [14]. The 4 frames are concatenated as the state s. The concatenation is defined as ϕ function. After being processed, the states are stored in stack D. The next section will introduce the whole process of DRL based on SARSA learning.

C. Deep SARSA learning

Given the number of video game n, the SARSA network should contain n outputs which represent n discrete state-action values, to interact with ALE. The current state s is processed by CNN to get the current state-action value Q_1, which is a n-dimension vector. Then the current action a is selected with ε-greedy algorithm. The reward r and the next state s' is observed. In order to estimate the current $Q(s,a)$, the next state-action value $Q(s',a')$ is obtained according to (4).

Here, when the next state s' is input into CNN, $Q(s',a')$ can be obtained. Then we define a label vector related to Q_1 being Q_2 which represents the target vector. The two vectors only have one different component. That is $r+\gamma Q(s',a') \to Q(s,a)$. Now the whole scheme of DRL based on SARSA learning is presented in Algorithm 1. It should be noted that during training, the next action a' for estimating the current state-action value is never greedy. On the contrary, there is a tiny probability that a random action is chosen.

Algorithm1 Deep Reinforcement Learning based on SARSA

initialize data stack D with size of N
and parameters of CNN θ
for episode=$1, M$ do
initialize state $s_1 = \{x_1\}$ and preprocess state $\phi_1 = \phi(s_1)$
  select a_1 with ε-greedy method
  for $t = 1, T$ do
    take action a_t, observe next state x_{t+1} and r_t,
    $\phi_{t+1} = \phi(s_{t+1})$
    store data $(\phi_t, a_t, r_t, \phi_{t+1})$ into stack D
    sample data from stack D
    select a' with ε-greedy method
    $$y_j = \begin{cases} r_j & if\ episode\ teminates\ at\ step\ j+1 \\ r_j + \gamma Q(\phi_{j+1}, a'; \theta) & otherwise \end{cases}$$
    according to (7), optimize the loss function
    $L_i(\theta_i)$
    $a_t \leftarrow a'$
  end for
end for

IV. EXPERIMENTS AND RESULTS

In this section, two simulation experiments will be presented to verify our algorithm. The two video games are from Atari 2600, called breakout and seaquest. Fig. 3 shows the images of the two games. The CNN contains 3 convolution layers and two full connected layers. All the settings in these two experiments are the same as DQN [14], except for the RL method. The discount factor is 0.99. Every 250 thousand steps, the agent is tested. Every testing episode are 125 thousand steps.

A. Breakout

In breakout, 5 basic actions including up, down, left, right and null are given. The operation image is like the left of Fig. 3. This game expects the agent to obtain as many scores as possible. The agent controls dam-board which can reflect the bullet. Once the bullet hits bricks in the top area, the agent gets 1 point. If the bullet falls down, the number of lives is subtracted 1 until the game is over.

Fig. 3 Two video games: breakout and seaquest.

Fig. 4 and Fig. 5 present the average score with deep SARSA learning and deep Q learning. We can see that at the end of the 20th epoch, deep SARSA learning reaches an average reward of about 100. By contrast, deep Q learning can reach about 170. We can conclude that in the early stage of training, deep SARSA learning converges slower than deep Q learning. However, after 30 epochs, deep SARSA learning gains higher average scores. In addition, deep SARSA learning converges more stably than deep Q learning.

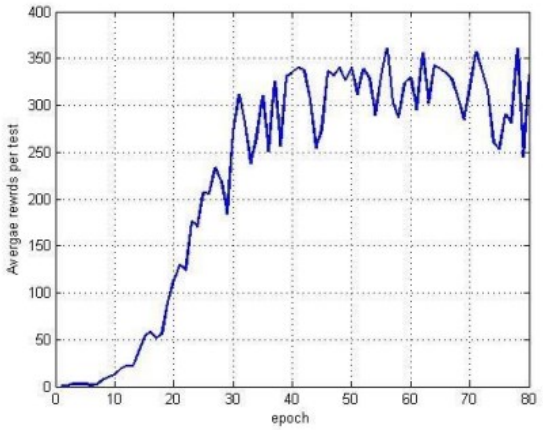

Fig. 4 Average score with deep SARSA learning in Breakout

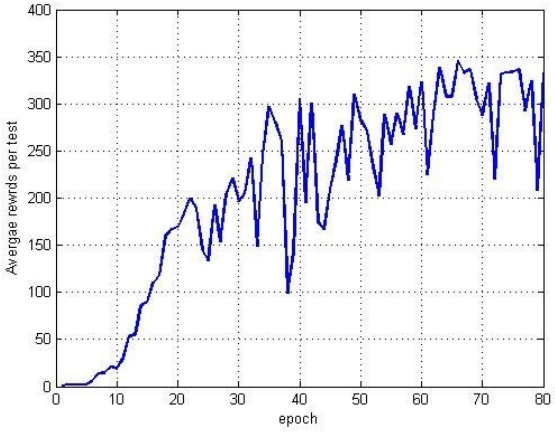

Fig. 5 Average score with deep Q learning in Breakout

The number of games during test with two algorithms is displayed in Figs. 6 and 7. It reflects the convergent trends of these algorithms. After training 20 epochs, deep SARSA learning can also converge to the equilibrium point at about 75. In deep Q learning, the equilibrium point is about 80.

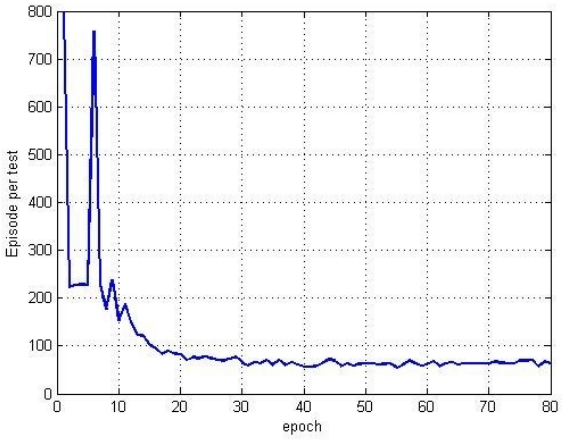

Fig. 6 Number of games during test with deep SARSA learning in Breakout

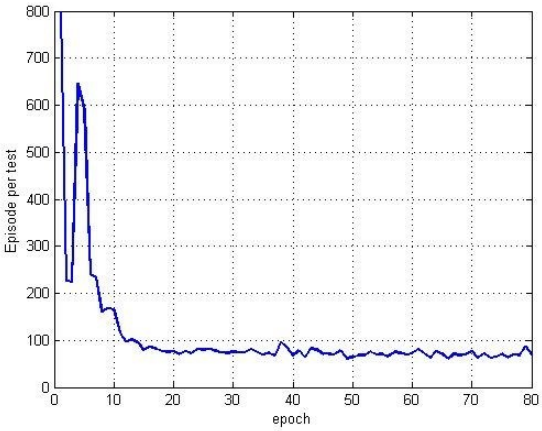

Fig. 7 Number of games during test with deep Q learning in Breakout

B. Seaquest

In seaquest, 5 basic actions are given including up, down, left, right and firing. The operation image is shown in the right of Fig. 3. This game expects that the agent should obtain as many scores as possible by saving divers and killing fish. The agent can control the submarine with five basic actions as mentioned above. Once the submarine saves the diver or kills fish, the agent gets 20 and 40 points. If the submarine runs into fish or the oxygen in the submarine is 0, the number of life drops 1 until the game is over. So if human play this game, the quantity of oxygen should also be taken into consideration.

Fig. 8 and Fig. 9 show the average score of deep SARSA learning and deep Q learning. We can see that the score of deep SARSA learning increases a little slower before the 10th epoch than deep Q learning. However, it will converge much faster after the 30th epoch. At last deep SARSA learning can gain about 5000 points while deep Q learning only gets 3700 points.

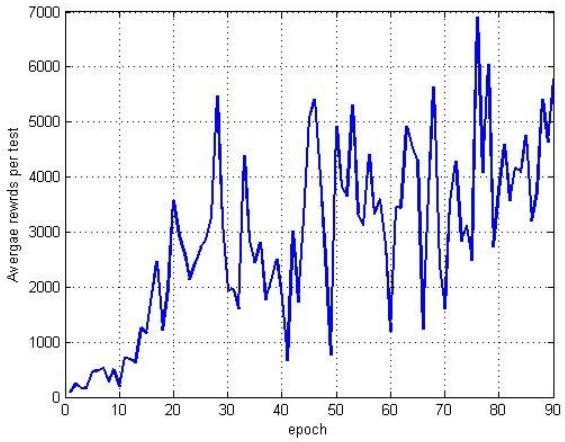

Fig. 8 Average score with deep SARSA learning in Seaquest

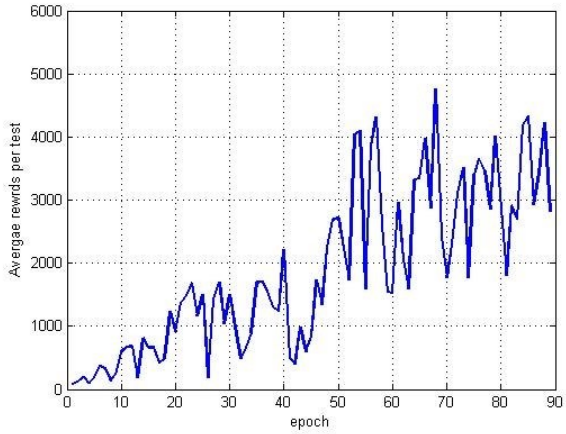

Fig. 9 Average score with deep Q learning in Seaquest

The number of games during test with two algorithms is shown in Fig. 10 and Fig. 11. It can also reflect the trend of DRL process. Deep SARSA learning even shows a smother process in this video game than deep Q learning.

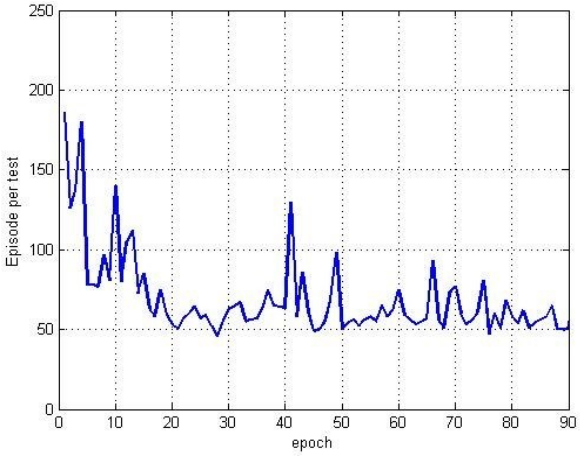

Fig. 10 Number of games during test with deep SARSA learning in Seaquest

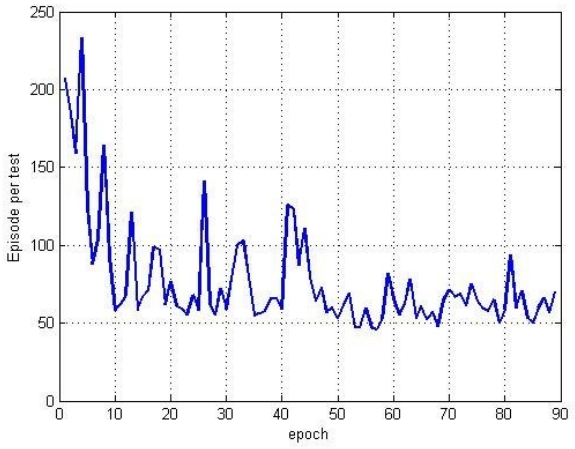

Fig. 11 Number of games during test with deep Q learning in Seaquest

V. CONCLUSION

In this paper, we introduce an on-policy method SARSA learning to DRL. SARSA learning has some advantages when being applied to decision making problems. It makes learning process more stable and is more suitable to some complicated systems. Given these facts, a new DRL algorithm based on SARSA, called deep SARSA learning, is proposed to solve the control problems of video games. Two simulation experiments are given to compare the performance of deep SARSA learning and deep Q learning. In Section 4, the results reveal that deep SARSA learning gains higher scores and faster convergence in breakout and seaquest than deep Q learning.

REFERENCES

[1] LeCun, Y., Y. Bengio, and G. Hinton, Deep learning. Nature, 2015, 521(7553): p. 436-444.

[2] LeCun, Y., Bottou L. and Bengio Y, Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998, 86(11): p. 2278-2324.

[3] Farabet, C., Couprie C, Najman L and LeCun Y, Scene parsing with multiscale feature learning, purity trees, and optimal covers. arXiv preprint arXiv:1202.2160, 2012.

[4] Sutton, R.S. and A.G. Barto, Introduction to reinforcement learning. 1998: MIT Press.

[5] Wang, F.Y., H. Zhang, and D. Liu, Adaptive dynamic programming: an introduction. IEEE Computational Intelligence Magazine, 2009, 4(2): p. 39-47.

[6] Zhao, D. and Y. Zhu, MEC--a near-optimal online reinforcement learning algorithm for continuous deterministic systems. IEEE Transactions on Neural Networks and Learning Systems, 2015, 26(2): 346-356.

[7] Zhu, Y., D. Zhao, and X. Li, Using reinforcement learning techniques to solve continuous-time non-linear optimal tracking problem without system dynamics. IET Control Theory & Applications, 2016, 10(12), 1339-1347.

[8] Zhu, Y. and D. Zhao, A data-based online reinforcement learning algorithm satisfying probably approximately correct principle. Neural Computing and Applications, 2015, 26(4): p. 775-787.

[9] Xia, Z. and D. Zhao, Online bayesian reinforcement learning by gaussian processes. IET Control Theory & Applications, 2016.10(12), 1331-1338.

[10] Lange, S. and M. Riedmiller. Deep auto-encoder neural networks in reinforcement learning. in The 2010 International Joint Conference on Neural Networks (IJCNN). 2010.

[11] Abtahi, F. and I. Fasel, Deep belief nets as function approximators for reinforcement learning, in Proceedings of IEEE ICDL-EPIROB. 2011.

[12] Arel, I., Deep Reinforcement Learning as Foundation for Artificial General Intelligence, in Theoretical Foundations of Artificial General Intelligence. 2012, Springer. p. 89-102.

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

[14] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. Nature, 2015, 518(7540): 529-533.

[15] Bellemare M G, Naddaf Y, Veness J, et al. The arcade learning environment: an evaluation platform for general agents. Journal of Artificial Intelligence Research, 2012, 47:253-279.

[16] Levine, S., Exploring deep and recurrent architectures for optimal control. arXiv preprint arXiv:1311.1761, 2013.

[17] Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search. Nature, 2016, 529(7587): 484-489.

[18] Defazio, A. and T. Graepel, A comparison of learning algorithms on the Arcade Learning Environment. arXiv preprint arXiv:1410.8620, 2014.

[19] Zeiler, M.D., ADADELTA: an adaptive learning rate method. arXiv preprint arXiv:1212.5701, 2012.

[20] Lin, L.J., Reinforcement learning for robots using neural networks. 1993, Technical report: DTIC Document.