Paper:

Experimental Study on Behavior Acquisition of Mobile Robot by Deep Q-Network

Hikaru Sasaki*, Tadashi Horiuchi, and Satoru Kato*****

*Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama-cho, Ikoma, Nara 630-0192, Japan
E-mail: sasaki.hikaru.rw3@is.naist.ac.jp
**Department of Control Engineering, National Institute of Technology, Matsue College
14-4 Nishi-ikuma, Matsue, Shimane 690-8518, Japan
E-mail: horiuchi@matsue-ct.ac.jp
***Department of Information Engineering, National Institute of Technology, Matsue College
14-4 Nishi-ikuma, Matsue, Shimane 690-8518, Japan
E-mail: kato@matsue-ct.ac.jp
[Received March 21, 2017; accepted July 21, 2017]

Deep Q-network (DQN) is one of the most famous methods of deep reinforcement learning. DQN approximates the action-value function using Convolutional Neural Network (CNN) and updates it using Q-learning. In this study, we applied DQN to robot behavior learning in a simulation environment. We constructed the simulation environment for a two-wheeled mobile robot using the robot simulation software, Webots. The mobile robot acquired good behavior such as avoiding walls and moving along a center line by learning from high-dimensional visual information supplied as input data. We propose a method that reuses the best target network so far when the learning performance suddenly falls. Moreover, we incorporate Profit Sharing method into DQN in order to accelerate learning. Through the simulation experiment, we confirmed that our method is effective.

Keywords: deep reinforcement learning, deep Q-network, mobile robot, behavior acquisition

1. Introduction

Recently, deep reinforcement learning, which combines deep learning and reinforcement learning has been receiving increased attention. Deep Q-network (DQN) [1, 2] is one of the most famous deep reinforcement learning methods. DQN proved its ability by enabling entities to acquire game playing skills superior to those of human experts in several games on the Atari 2600. Deal with high-dimensional image data as input data is difficult in the conventional reinforcement learning. In contrast, DQN has remarkable features to directly learn action policies from such high-dimensional image data by using Convolutional Neural Network (CNN) [3], which is one of the major approaches in deep learning.

Our focus is on DQN because it uses CNN, which is regarded as a kind of artificial visual cortex system and a "vision system" technology [4]. In animals, the evolution of eyes enabled a higher possibility of survival in the severe world [5]. We focus on vison-based robot learning because visuomotor learning is very important considering that the evolution of a vision system in animals enabled the acquisition of survival behavior. Breakthroughs in this domain will have significant impacts in the field of robot learning.

Moreover, vision-based robots have several advantages over non-vision-based robots that use other sensors. For example, it is possible to use a camera inside buildings, whereas whereas GPS sensors are not applicable in indoor environments. A vision system is a much wider ranging sensor than a distance sensor, but it is not as accurate. However, it is more suitable for recognizing the surrounding environment of a robot than a gyro sensor or an accelerometer.

It has been demonstrated that four-legged robots and snake-like robots with web camera and the palm-size computer Raspberry Pi can acquire behavior that enables them to reach targets while avoiding obstacles by using CPG (Central Pattern Generator) and reinforcement learning [6, 7]. However, in those cases, human designed the discrete state space based on images obtained by a web camera on the robot.

In this study, we applied DQN to robot behavior learning to make it unnecessary for humans to design the state space. In this way, a mobile robot learned to acquire good behavior such as avoiding walls and moving along a center line by using high-dimensional visual information as input data in the simulation environment. We constructed a simulation environment for the mobile robot using the robot simulation software, Webots. As one of the first steps toward vision-based deep reinforcement learning for mobile robots, we applied DQN to the two-wheeled mobile robot because it is easier to construct a simulation model of a two-wheeled robot than a four-legged robot or a snake-like robot. The mobile robot has a camera on

top of its body and DQN uses the image data obtained by the camera as input data. First, we show that the original DQN does not have good learning performance in our robot navigation problem. Then, we propose a modified DQN method that reuses the best target network so far when the performance of learning suddenly decreases. Moreover, we realize acceleration of learning by incorporating Profit Sharing method into DQN. Finally, we confirm that DQN has good generalization ability by evaluation in a test environment that differs from the training environment.

The remainder of this paper is organized as follows. Section 2 presents several related studies in which DQN is applied to robot learning. Section 3 gives an overview of the DQN, including its algorithm. Section 4 explains the robot navigation problem and how DQN is applied to the problem. Section 5 introduces our proposed modification of DQN for robot learning. Section 6 presents experimental results and discussions. Section 7 concludes and outlines future work.

2. Related Work

There are several related studies in which DQN is applied to robot learning. For example, Tai et al. [8] presented an approach that realizes mobile robot exploration through CNN-based reinforcement learning (DQN) using the Gazebo simulation environment. The robot used was a Tutlebot, a two-wheeled robot with a Kinect sensor. They used the depth images as input and the original DQN itself as the learning system. However, they could not get good test performance in the simulation experiment. Their robot and target task are similar to those in our study. However, we used a normal camera (not a depth camera) and our modified DQN method had very good learning and test performance in our simulation experiment.

Zhang et al. [9] presented a DQN-based learning system for a target reaching task. The robot used was a Baxter's robot arm. Further, they used a 2D target reaching simulator to train DQN in several training scenarios with original DQN as the learning system. As their robot and target task differ from ours, it is not possible to compare them directly with our study. However, we surmise that our method has an advantage in that it can avoid unstable learning by reuse of the most recent and best target network so far when the learning performance suddenly decreases.

Amarjyoti [10] reviewed various approaches and algorithms applied to reinforcement learning in robotic manipulation tasks. He reviewed the original DQN, Double DQN, and the continuous action space algorithms, including Normalized Advantage Functions (NAFs), the stochastic policy gradient and stochastic actor-critic method, and the deterministic policy gradient method. We consider methods such as Double DQN, stochastic policy gradient and stochastic actor-critic methods promising for the robot navigation problem will investigate the effect of these methods in future work.

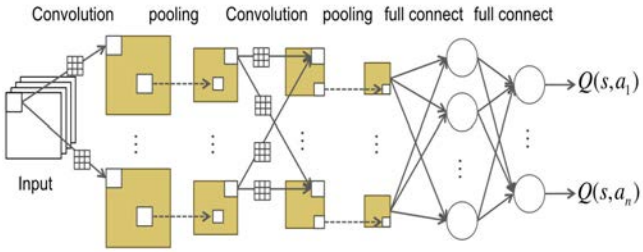


Fig. 1. Structure of DQN.

for episode $= 1, M$ **do**
  Observe initial state s_1
  for $t = 1, T$ **do**
    Determine action a_t using action selection
    Execute action a_t and observe reward r_t and next state s_{t+1}
    Store the transition (s_t, a_t, r_t, s_{t+1}) in data set D
    Sample randomly minibatch of transitions from D
    Calculate the loss function
    Update the parameters of learning network $Q(s,a)$ using RMSProp method
    Update target network $\hat{Q} = Q$ every C steps
  end for
end for

Fig. 2. Algorithm of DQN.

Silver [11], Lillicrap et al. [12], and Mnih et al. [13] proposed various algorithms, including the above methods, and applied them to various simulated problems, such as Atari 2600 games, TORCS car racing simulator, MuJoCo physics simulator (cart-pole swing-up task, reaching task, locomotion task, etc.), and Labyrinth simulation environment. However, they did not use a mobile robot for the robot navigation problem. In contrast, we applied the modified DQN method to the mobile robot navigation problem and are currently in the process of applying it to a real mobile robot with a web camera and Raspberry Pi computer.

3. Deep Q-Network (DQN)

DQN uses Q-learning algorithm [14] one of the most widely-used reinforcement learning methods and CNN to approximate the action-value function called the Q-function. **Fig. 1** shows the structure of DQN.

Reinforcement learning is known to be sometimes unstable or even to diverge when a nonlinear function approximator such as neural network is used to represent the action-value function (Q-function). There are several reasons for this instability. They include 1) the correlations present in the sequence of state observations, s_t and s_{t+1}, and 2) the fact that small updates of Q-value may significantly change the policy π and there-

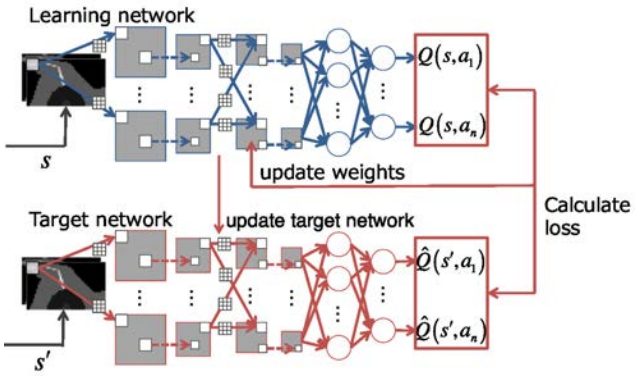


Fig. 3. Relationship between learning network and target network in DQN.

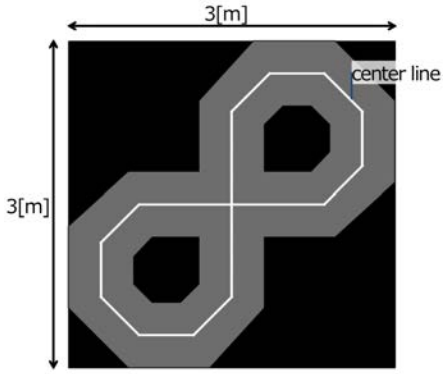
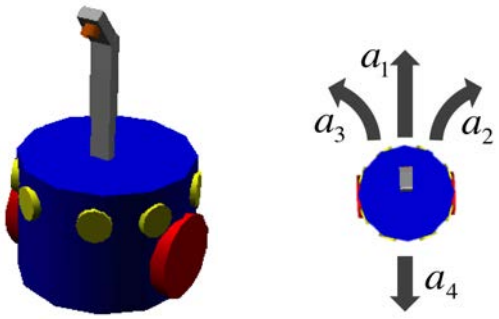


Fig. 4. Problem environment.



(a) Two-wheeled mobile robot (b) Candidate actions of robot

Fig. 5. Two-wheeled mobile robot and its candidate actions.

fore change the data distribution. To overcome the above problems, DQN uses a method called experience replay. To perform experience replay, the agent's experiences $e_t = (s_t, a_t, r_t, s_{t+1})$ at each time step t are stored in the data set $D = \{e_1, ..., e_t\}$ also called replay memory. During learning, Q-learning updates are applied on samples of experience (s_j, a_j, r_j, s_{j+1}) drawn uniformly at random from data set D.

Figure 2 shows the algorithm of deep Q-learning with experience replay. In **Fig. 2**, M signifies the maximum number of episodes, T the maximum number of action steps per episode, and C the interval within which to update the target network parameters (the target network parameters are updated every C steps). The Q-learning update at iteration i uses the following loss function:

$$L_i(\theta_i) = E[(r + \gamma \max_{a'} \hat{Q}(s', a'; \theta_i^-) - Q(s, a; \theta_i))^2]$$

where r is the reward, γ the discount factor, θ_i the network parameters of the Q-network at iteration i, and θ_i^- the network parameters used to compute the target $y_j = r + \gamma \max_{a'} \hat{Q}(s', a'; \theta_i^-)$ at iteration i. Here, the iteration i indicates the i-th update of the parameters of the Q-network. Further, time step t signifies the t-th action step of the robot in each episode. **Fig. 3** illustrates the relationship between the learning network Q and the target network $\hat{Q}$. The loss function is minimized by using RMSProp [15] a stochastic gradient descent method.

4. Robot Learning by DQN

4.1. Problem Setting

In this study, we applied DQN to robot behavior learning in a simulation environment. We realized behavior acquisition of the two-wheeled mobile robot in the simulation environment. For this purpose, we used Cyberbotics's robot simulator Webots.

The target task for the robot was acquisition of the behavior of a two-wheeled mobile robot to move without colliding with the walls in the problem environment shown in **Fig. 4**. **Fig. 5(a)** depicts the appearance of the two-wheeled mobile robot. The size of the problem environment was 3 m × 3 m and the width of the road was 60 cm. The center line was painted white in the center of the road. The road had the shape of a slanted number "8" as shown in **Fig. 4**, and the height of the wall was 15 cm, which is a little higher than the robot itself.

4.2. The Two-Wheeled Mobile Robot

The specifications of the two-wheeled mobile robot are shown in **Table 1**. A camera on top of the robot gives it a front view. This camera outputs 50 × 30 pixels images. **Fig. 6** presents four sample camera images of the robot. In the images, the walls are black, the floor is gray, the center line is white, and the body of robot is blue. We placed the camera on top of the robot in order for it to see a portion of its own body. In the next section, we discuss the experiment in which we attached the camera in such a manner that the robot could not see its own body and only could see the road, wall, and center line.

The robot also has distance sensors in eight directions, which enable it to measure the distance between itself and the wall. The measurable range of each sensor is between 1.0 cm to 70 cm.

Table 1. Specifications of the two-wheeled mobile robot.

Diameter of body	16 cm
Height	12 cm
Diameter of wheel	5 cm
Camera	1 direction
Distance sensors	8 directions

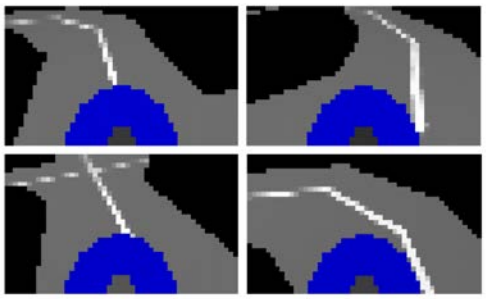


Fig. 6. Sample camera images of the robot.

4.3. Definition of Action, State, and Reward

The robot takes an action at each time step from among the following four candidates:

$$a_t = \begin{cases} a_1 & \text{(go straight)} \\ a_2 & \text{(turn right)} \\ a_3 & \text{(turn left)} \\ a_4 & \text{(go backward).} \end{cases}$$

The action "go straight" is realized by rotating both wheels clockwise at the same speed, which causes the robot to move forward about 5.0 cm by this action. The action "turn right" is realized by rotating the left wheel clockwise faster than the right wheel. This action causes, the robot to move approximately 7.0 cm and turn approximately 13.2 deg in the clockwise direction. The action "turn left" is realized by moving the two wheels with motion opposite to those for "turn right." The action "go backward" is also realized by the opposite motions of the two wheels to those for "go straight." Taking an action at each time step represents one step action in this study.

The state observation is defined using the camera images of the robot. Suppose the camera image at time step t is x_t, we get $\phi_t = \phi(x_t)$ by applying the preprocessing function ϕ. Here, the preprocessing function transforms the RGB color image to a gray-scale image. The state s_t is defined by using the most recent two frames ϕ_t, ϕ_{t-1}. Hence, $s_t = \{\phi_t, \phi_{t-1}\}$.

The reward r_t at time step t is defined by using the values of the eight distance sensors of the robot and the action selected by the robot as follows:

$$r_t = \begin{cases} 10 & \text{(when robot is far away from the wall)} \\ -10 & \text{(when robot selects the backward action)} \\ -20 & \text{(when robot collides to the wall)} \\ 0 & \text{(otherwise).} \end{cases}$$

The structure of CNN is as follows: there are two hidden layers each of which has 8×8 weight filters and 2×2 max-pooling. The number of weight filters is 32 both in the first convolution layer and in the second convolution layer. The inputs are two gray-scale images 50×30 pixels in size. The number of nodes in the hidden layer of the classification part is 256. The number of nodes in the output layer is four, which is equal to the number of action candidates of the robot.

5. Modification of DQN for Robot Learning

5.1. Reuse of the Best Target Network

In the original DQN, shown in **Fig. 2**, every C steps, we clone the network Q to obtain the target network $\hat{Q}$ and we use $\hat{Q}$ to generate the targets y_j $(= r + \gamma \max_{a'} \hat{Q}(s', a'; \theta_i^-))$ for the following C updates to Q. This makes the algorithm more stable than standard Q-learning. Generating targets y_j using an older set of network parameters adds a delay between the time an update to Q is made and the time the update affects targets y_j, making oscillations much more unlikely.

However, in our robot navigation problem, the original DQN showed unstable learning performance. In this study, we proposed a modified DQN method that stores the best target network parameters so far and replaces the target network $\hat{Q}$ to the best target network so far $\hat{Q}_{best}$, when the performance of DQN using the updated target network $\hat{Q}$ suddenly decreases in the performance evaluation (in the test run phase). This is based on our consideration that the performance of the target network $\hat{Q}$ does not always increase monotonically and sometimes suddenly falls in our robot navigation problem.

5.2. DQN with Profit Sharing Method

Profit Sharing method [16] is an "exploitation-oriented" reinforcement learning method that distributes the terminal reward all at once to the previous state-action pairs for the past several steps. Horiuchi et al. [17] proposed Q-PSP Learning, which incorporates the idea of Profit Sharing method into Q-learning, which is an "exploration-oriented" reinforcement learning method, in order to combine the merits of the two approaches. Miyazaki [18] proposed DQNwithPS to accelerate the learning of DQN.

In this study, we also realized acceleration of learning by incorporating Profit Sharing method into DQN. To accomplish this, we stored the set of transitions $e_t = (s_t, a_t, r_t, s_{t+1})$ for the latest K steps, where K indicates the maximum number of steps to distribute the reward all at once. Every time the agent acquires the terminal reward r_t, the immediate reward r_{t-i} at time $t-i$ is rewritten by the following decreasing geometric series function:

$$r_{t-i} = \alpha^i r_t \quad (i = 1 \sim K) \quad \ldots\ldots\ldots\ldots (1)$$

where α indicates the damping factor $(0 < \alpha < 1)$. Immediate reward r_{t-i} of the latest K steps experience before the agent acquires the terminal reward r_t, is rewritten by

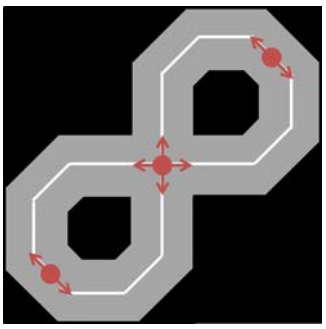


Fig. 7. Starting positions and directions of the mobile robot in the test run phase.

the above function, and then the rewritten K steps experience are stored in the replay memory D for the learning of DQN. Random sampling of a minibatch of transitions from replay memory D is adopted just as in the original DQN.

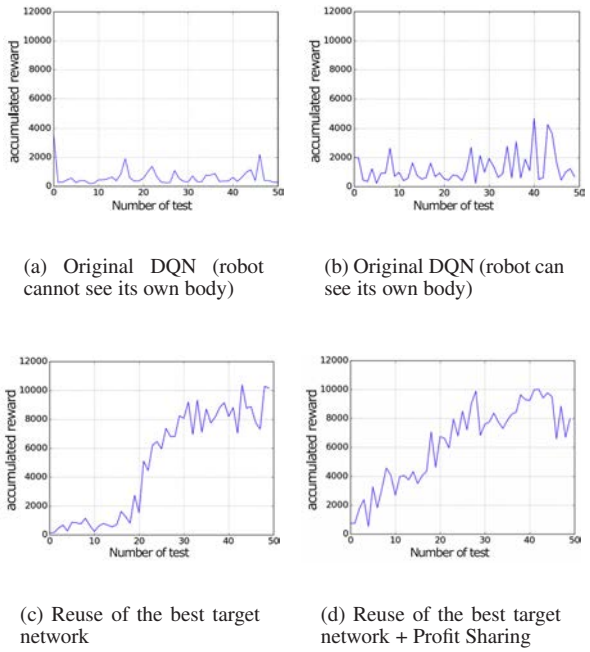
6. Experiments

6.1. Experiment 1 (Modification of DQN)

6.1.1. Experimental Method

We executed the behavior learning experiment of the mobile robot for the navigation problem as shown in the previous sections. In the learning phase, we adopted the ε-greedy method as the action selection method and the value of ε was fixed at 0.2. The robot started from the central point of **Fig. 7** and randomly selected the initial direction among the four directions (up/down/left/right). On collision with the wall, the robot executed the backward actions until it approximately reached the center line and then restarted in a randomly selected direction. During the initial $10,000$ steps, the robot only collected experience data to use in experience replay. The size of replay memory D was $100,000$ and minibatch size was 32. After the initial $10,000$ steps, we evaluated the learning performance as test run phase every $5,000$ learning steps. The target network update interval was $5,000$. In the test phase, the starting point of the robot was randomly selected among the three positions and eight directions, as shown in **Fig. 7**. In the test phase, we used the greedy method as the action selection method.

To realize stable learning, we adopted the modified DQN method, which reuses the best target network so far when the learning performance of learning suddenly decreases. In this experiment, we judged that the learning performance suddenly decreases when the acquired reward in the test phase is less than 30% of the highest acquired reward in the previous test runs up to that point. Profit Sharing method was executed only for the penalty reward -20 (when the robot collided with the wall). The values of the parameters in Profit Sharing method were as follows: the value of K (maximum number of steps to distribute the reward at once) $= 5$ and the value of damping factor $\alpha = 0.7$. The value of K was set to five because the number of action steps taken by the robot from around the center line to the wall collision was approximately five.



(a) Original DQN (robot cannot see its own body)

(b) Original DQN (robot can see its own body)

(c) Reuse of the best target network

(d) Reuse of the best target network + Profit Sharing

Fig. 8. Change of acquired reward in test run by each method.

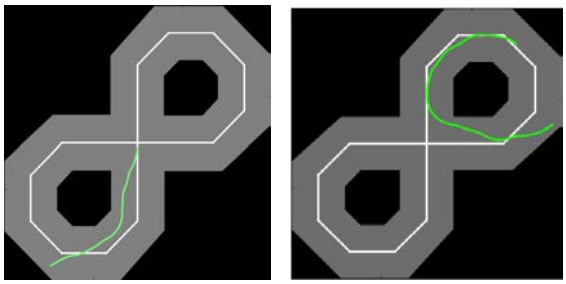
In this section, we compare the following four methods: (a) original DQN (for the case where the robot cannot see its own body), (b) original DQN (for the case where the robot can see its own body), (c) modified method with reuse of the best target network so far when the learning performance suddenly decreases (for the case where the robot can see its own body), (d) modified method with reuse of the best target network and Profit Sharing method (for the case where the robot can see its own body).

6.1.2. Experimental Results

We executed five sets of simulations for each of methods (a)–(d) with different random seeds. **Fig. 8** illustrates the change in the average of the accumulated acquired reward in each test run. In each graph, the horizontal axis represents the number of test runs and the vertical axis the average of the accumulated acquired reward in each test run.

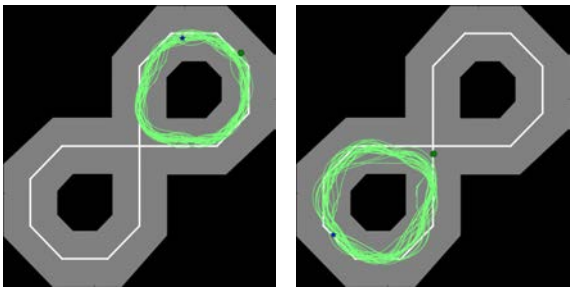
Compared with method (a) original DQN (for the case where the robot cannot see its own body) and method (b) original DQN (for the case where the robot can see its own body), it is clear that method (b) has better learning performance. This is because it is important for the robot to know how its body collides with the wall and the positional relationship between its body and the wall. We think that method (b) makes it easier than method (a) for the mobile robot to know the locational situation in detail, including its orientation and its proximity to the wall. However, the learning performance of the original DQN is relatively low even in method (b).

In contrast, method (c) shows good learning performance by using our modified method with reuse of tar-



(a) Original DQN (robot cannot see its own body)

(b) Original DQN (robot can see its own body)



(c) Reuse of the best target network

(d) Reuse of the best target network + Profit Sharing

Fig. 9. Examples of behavior trajectory acquired by each method.

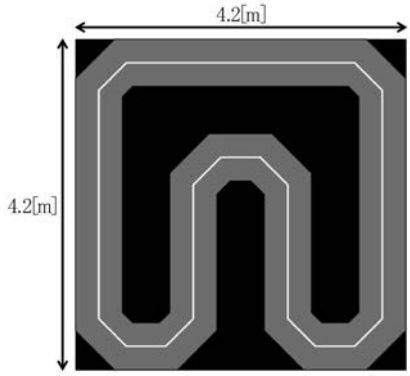


Fig. 10. Test environment.

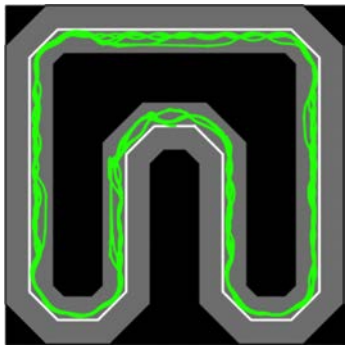


Fig. 11. Example of behavior trajectory in the test environment.

get network when the learning performance suddenly decreases. Hence, we believe that method (b), in which the robot can see its own body, is still not so adequate because of unstable learning performance. Using the modified method with reuse of the best target network so far combined with method (b), method (c) shows much better performance than method (b). Moreover, method (d) has early increases in learning performance by incorporating Profit Sharing method into the method (c). In **Fig. 8**, focusing on the early stage of learning before 20 tests, method (d) shows much faster and greater increase in accumulated reward than method (c). This is the main effect of incorporating the Profit Sharing method into the method (c). After the middle stage of learning, the performance of method (d) is similar to that of method (c). We think it is necessary to investigate a method that uses Profit Sharing method only in the early stage of learning.

Figure 9 shows examples of the trajectories of robot behaviors acquired by each method after learning. In methods (a) and (b), both of which use the original DQN, the robot collides with the wall relatively soon after starting. However, in methods (c) and (d), both of which reuse the target network in DQN, the robot acquires the habit of moving away from the wall, and consequently, it tends to move near to the center area of the road, as shown in **Figs. 9(c)** and **9(d)**.

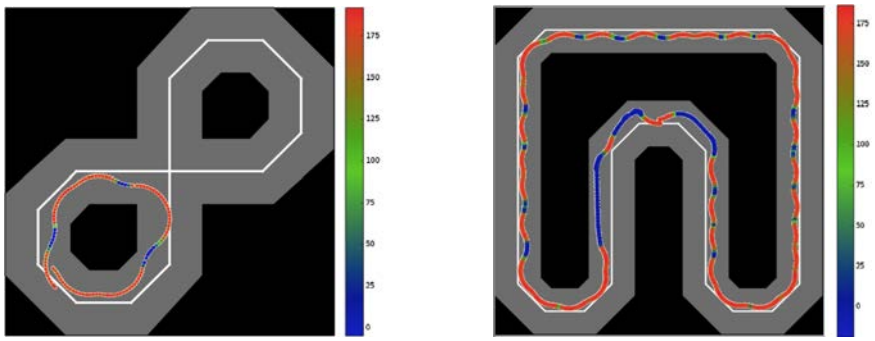
6.2. Experiment 2 (Generalization Ability of DQN)

6.2.1. Experimental Method

We confirmed the good generalization ability of the robot with the DQN evaluating it in a test environment that differed from the training environment. This kind of evaluation for robot learning was not carried out in the original DQN study. In this experiment, we applied the DQN obtained by learning in the previous experiment to the new test environment illustrated in **Fig. 10**. In this test environment, the road was concave instead of being shaped like a slanted digit "8" and it was necessary for the robot to have the ability to turn left and right and to go straight for long distances. After training DQN using method (d) in the previous section for the training environment shown in **Fig. 4**, we applied the learned DQN to the test environment.

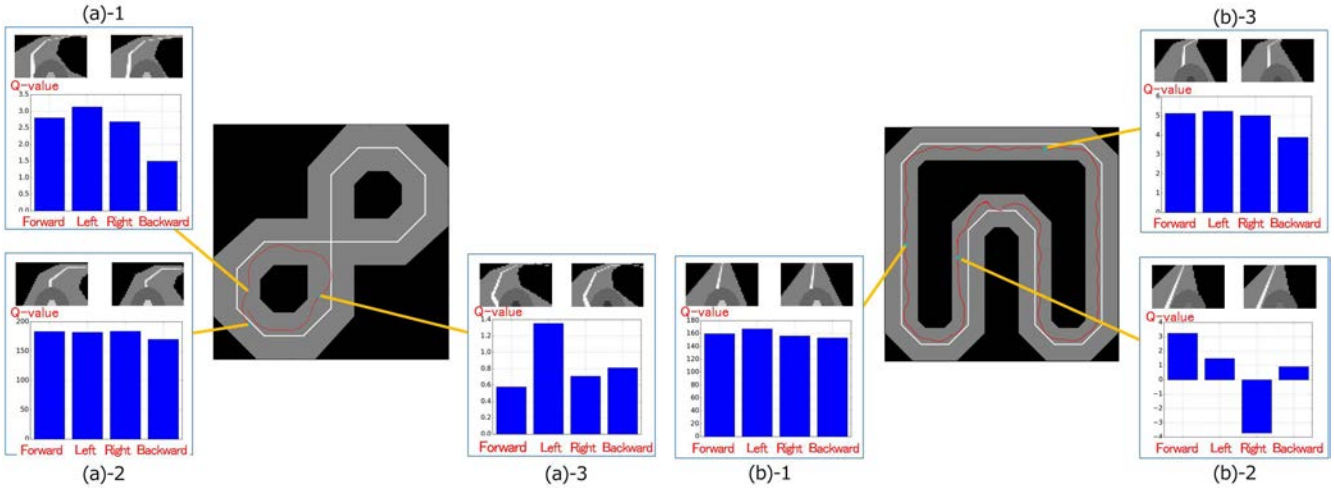
6.2.2. Experimental Results

Figure 11 illustrates an example of the trajectory of the robot on applying the learned DQN to the new test environment. From this figure, it can be seen that the robot was able to avoid the wall and turn left and right appropriately for the new environment, which is different from the training environment. In the previous experiment, the behavior trajectory acquired in the training environment was like drawing a circle consisting of the action to go straight and either the action to turn left or right. However, in this test environment, the robot could turn left and right and



(a) Training environment (b) Test environment

Fig. 12. Change of state-value along the behavior trajectory of the robot.



(a) Training environment (b) Test environment

Fig. 13. Examples of action-value in several situations.

go straight for long distances. Therefore, we confirmed that the robot with DQN has good generalization ability.

6.3. Analysis of the Acquired Knowledge

Figure 12 illustrates the acquired state-value by DQN learning along the behavior trajectory of the robot in both the training environment shown in **Fig. 4** and the test environment shown in **Fig. 10**. Here, the state-value is computed by the following equation:

$$V(s) = \max_a Q(s,a). \quad \ldots \quad (2)$$

From **Fig. 12**, it is clear that the robot tends to select actions that have high state-value. However, there are also low state-value actions which mean non-optimal actions such as approaching the wall.

Figure 13 indicates the action-value $Q(s,a)$ when the robot moves in each environment using the learned DQN. In each bar graph, the vertical axis denotes the action-value $Q(s,a)$ in each situation (location). Above each bar graph, the corresponding camera images of the robot are shown in each situation. Focusing on each situation, the action-value tends to be low when the robot is close to the wall or facing the wall. For example, it can be seen that the action-value for turn Right is very low (large negative) in **Fig. 13(b)-2**, because the action of turn Right causes the collision to the wall in this situation. On the other hand, the action-value is high when the robot is far from the wall, as shown in **Figs. 13(a)-2**, **13(b)-1**, and **13(b)-3**. Moreover, the action of the robot to move away from the wall has high action-value. For example, the action-value for turn Left is higher than those of other actions in this situation in **Figs. 13(a)-3** and **13(a)-1**, because the robot gets close to the center of the road (far away from the wall) and the robot gets higher reward than other actions. The curve of the road also affects the action-value. Both the actions turn Left and turn Right have high action-value in **Figs. 13(a)-1**, **13(a)-3**, and **13(b)-3**, where there is a right curve a little ahead of the current robot position. On the other hand, the action of turn Right has minus action-

value in **Fig. 13(b)-2**, where the road is straight at this robot position. Therefore, we can conclude that appropriate action knowledge is acquired through learning by our DQN method.

7. Conclusions

In this study, we applied DQN to vision-based mobile robot behavior learning and enabled a mobile robot to acquire good behavior such as avoiding walls and moving along a center line using high-dimensional visual information as input data in a simulation environment. We showed that the original DQN does not have good learning performance in our robot navigation problem and proposed a modified DQN method that reuses the best target network so far when the learning performance suddenly decreases. Further, we realized accelerated learning by incorporating Profit Sharing method into DQN. We also confirmed that DQN has good generalization ability by evaluating it in a test environment that differed from the training environment.

Our experimental results are preliminary. Therefore, it is necessary to conduct more thorough experiments and to investigate the effect of the proposed method, including dependency on the starting point in the test run, and decrease in the accumulated reward after 30 and 45 tests in method (d). Moreover, we plan to apply our proposed method to a real mobile robot. Consequently, we are developing a two-wheeled mobile robot with a web camera and the palm-size computer Raspberry Pi, and also preparing the a problem environment similar to the simulation environment. In this manner, we aim to realize robot behavior learning based on visual information using our DQN method in the real robot environment.

Acknowledgements

This research was partially supported by JSPS KAKENHI Grant Number 15K00355 from the Japan Society for the Promotion of Science.

References:

[1] V. Mnih et al., "Playing Atari with Deep Reinforcement Learning," Proc. of NIPS 2013 Deep Learning Workshop, 2013.

[2] V. Mnih et al., "Human-level Control through Deep Reinforcement Learning," Nature, Vol.518, pp. 529-533, 2015.

[3] Y. LeCun et al., "Gradient-based Learning applied to Document Recognition," Proc. of the IEEE, Vol.86, No.11, pp. 2278-2324, 1998.

[4] Y. Matsuo, "Expectation of Robot Field from Artificial Intelligence Field," J. of the Robotics Society of Japan, Vol.35, No.3, pp. 2-7, 2017 (in Japanese).

[5] E. Yong, "Inside the Eye: Nature's Most Exquisite Creation," National Geographic Magazine, Vol.22, No.2, 2016.

[6] S. Yoshigi, T. Mikami, and T. Horiuchi, "Behavior Acquisition of Autonomous Four-Legged Robot with CPG and Reinforcement Learning," Proc. of 2016 Annual Conf. of Electronics, Information and Systems Society, I.E.E. of Japan, pp. 1311-1312, 2016 (in Japanese).

[7] N. Nagami, R. Kishimoto, and T. Horiuchi, "Acquisition of Goal-Oriented Behavior for Snake-Like Robot by CPG and Reinforcement Learning," J. of Japan Society for Fuzzy Theory and Intelligent Informatics, Vol.29, No.2, pp. 551-557, 2017 (in Japanese).

[8] L. Tai and M. Liu, "Mobile Robots Exploration through CNN based Reinforcement Learning," Robotics and Biomimetics, Vol.3, No.24, 2016.

[9] F. Zhang et al., "Towards Vision-Based Deep Reinforcement Learning for Robotic Motion Control," arXiv:1511.03791, 13 Nov 2015.

[10] S. Amarjyoti, "Deep reinforcement learning for robotic manipulation – the state of the art," arXiv:1701.08878, 31 Jan 2017.

[11] D. Silver, "Tutorial: Deep Reinforcement Learning," Proc. of the 33rd Int. Conf. on Machine Learning (ICML 2016), 2016.

[12] T. P. Lillicrap et al., "Continuous Control with Deep Reinforcement Learning," arXiv:1509.02971, 29 Feb 2016.

[13] V. Mnih et al., "Asynchronous Methods for Deep Reinforcement Learning," arXiv:1602.01783, 16 Jun 2016.

[14] C. J. Watkins and P. Dayan, "Technical Note: Q-Learning," Machine Learning, Vol.8, pp. 279-292, 1992.

[15] A. Graves, "Generating Sequences with Recurrent Neural Networks," arXiv:1308.0850, 2013.

[16] K. Miyazaki, H. Kimura, and S. Kobayashi, "Theory and Applications of Reinforcement Learning Based on Profit Sharing," Trans. of the Japanese Society for Artificial Intelligence, Vol.14, No.5, pp. 800-807, 1999 (in Japanese).

[17] T. Horiuchi, A. Fujino, O. Katai, and T. Sawaragi, "Q-PSP Learning: An Exploitation-Oriented Q-Learning Algorithm and Its Applications," Trans. of the Society of Instrument and Control Engineers, Vol.35, No.5, pp. 645-653, 1999 (in Japanese).

[18] K. Miyazaki, "Experimental Results of Exploitation-oriented Learning with Deep Learning," The papers of Technical Meeting on Systems, I.E.E. of Japan, ST-16-049, pp. 41-46, 2016 (in Japanese).

Name:
Hikaru Sasaki

Affiliation:
Graduate Student, Graduate School of Information Science, Nara Institute of Science and Technology

Address:
8916-5 Takayama-cho, Ikoma, Nara 630-0192, Japan

Brief Biographical History:

2010- Department of Control Engineering, National Institute of Technology, Matsue College

2015- Advanced Engineering Faculty, National Institute of Technology, Matsue College

2017- Graduate School of Information Science, Nara Institute of Science and Technology

Main Works:

• H. Sasaki, T. Horiuchi, and S. Kato, "A Study on Behavior Acquisition of Mobile Robot by Deep Q-Network," Proc. of the 11th Int. Conf. on Innovative Computing, Information and Control, SS05-06, 2016.

Membership in Academic Societies:

• The Institute of Electrical Engineers of Japan (IEEJ)

Name:
Tadashi Horiuchi

Affiliation:
Professor, Department of Control Engineering, National Institute of Technology, Matsue College

Address:
14-4 Nishi-ikuma, Matsue, Shimane 690-8518, Japan

Brief Biographical History:
1997- Research Associate, Institute of Scientific and Industrial Research, Osaka University
2001- Associate Professor, National Institute of Technology, Matsue College
2017- Professor, National Institute of Technology, Matsue College

Main Works:
• Y. Ishikura, R. Kishimoto, and T. Horiuchi, "Acquisition of Goal-Oriented Behaviors for Multi-Legged Robots by CPG and Reinforcement Learning," IEEJ Trans. on Electronics, Information and Systems, Vol.136, No.3 pp. 333-339, 2016.

Membership in Academic Societies:
• The Society of Instrument and Control Engineers (SICE)
• The Institute of Electrical Engineers of Japan (IEEJ)
• The Japan Society for Fuzzy Theory and Intelligent Informatics (SOFT)

Name:
Satoru Kato

Affiliation:
Associate Professor, Department of Information Engineering, National Institute of Technology, Matsue College

Address:
14-4 Nishi-ikuma, Matsue, Shimane 690-8518, Japan

Brief Biographical History:
2001- Research Associate, National Institute of Technology, Matsue College
2009- Lecturer, National Institute of Technology, Matsue College
2012- Associate Professor, National Institute of Technology, Matsue College

Main Works:
• S. Kato, T. Horiuchi, and Y. Itoh, "A Study on Clustering Method by Self-Organizing Map and Information Criteria," J. of Japan Society for Fuzzy Theory and Intelligent Informatics, Vol.21, No.4, pp. 452-460, 2009.

Membership in Academic Societies:
• The Information Processing Society of Japan (IPSJ)
• The Institute of Electrical Engineers of Japan (IEEJ)
• The Japan Society for Fuzzy Theory and Intelligent Informatics (SOFT)