Discovery of Pattern Meaning from Delayed Rewards by Reinforcement Learning with a Recurrent Neural Network

Katsunari Shibata and Hiroki Utsunomiya

Abstract— **In this paper, by the combination of reinforcement learning and a recurrent neural network, the authors try to provide an explanation for the question: why humans can discover the meaning of patterns and acquire appropriate behaviors based on it. Using a system with a real movable camera, it is demonstrated in a simple task in which the system discovers pattern meaning from delayed rewards by reinforcement learning with a recurrent neural network. When the system moves its camera to the direction of an arrow presented on a display, it can get a reward. One kind of arrow is chosen randomly among four kinds at each episode, and the input of the network is 1,560 visual signals from the camera. After learning, the system could move its camera to the arrow direction. It was found that some hidden neurons represented the arrow direction not depending on the presented arrow pattern and kept it after the arrow disappeared from the image, even though no arrow was seen when it was rewarded and no one told the system that the arrow direction is important to get the reward. Generalization to some new arrow patterns and associative memory function also can be seen to some extent.**

I. Introduction

Humans can read a sign or symbol from various images, understand its meaning, memorize it, and use it to generate appropriate actions. For example, let us consider the arrow problem that is picked up in the part of experiment in this paper. When we see an arrow on a sign, we guess that it means "go to the pointing direction" or "see the pointing direction", and then actually go to or see the direction. When the sign goes out of our visual field, the arrow direction is memorized, and we can act according to the memorized arrow direction.

When we want a robot to behave like that, we usually give a program to recognize the arrow direction, to memorize the recognized result and to behave according to it. In this case, the fundamental problem: "Why we humans can discover the meaning of arrows, and get to memorize it and to behave appropriately", is put aside. However, from the viewpoint of developing more flexible and intelligent robots, it must be important to tackle the fundamental problem: "how does a robot discover the meaning of arrows by itself" rather than to make a robot go to the arrow direction according to a given program. Brooks also mentioned that this "abstraction" is the essence of intelligence and the hard part of the problems being solved. He pointed out that usually the way of abstraction is given and the process after that is focused, but actually, abstraction itself is the essence of intelligence[1]

The authors have shown that by the combination of reinforcement learning and a neural network, through the learning of behaviors to get rewards and to avoid punishments, various necessary functions for the task, such as recognition or memory, emerge in the network[2]. If it is possible for humans to discover through learning that the arrows have a meaning and the meaning lies in their directions, the basis of learning that can be thought of most easily must be learning from experiences: when we turned to the direction of the arrow, we met something good. Of course, it can be considered that such knowledge is transferred from the others, but in this case, another similar problem arises; why we can get to understand what the others tell us.

Then in this paper, a system with an actual movable camera learns appropriate behaviors by reinforcement learning with a recurrent neural network (RNN) in the environment where an arrow pattern is presented and when the camera continues to move to the arrow direction even after it disappers, it reaches a goal state and gets a reward. It is verified (1) whether the RNN discovers that the arrow direction is important among 1,560 visual signals, (2) whether it acquires a way to recognize the arrow direction from the 1,560 signals, (3) whether it gets to keep the direction after the arrow disappears and to get the reward finally.

Some researches in which a RNN is used in reinforcement learning have been done already[3][4][5][6][7]. Especially, the pioneering work of Bakker et al. showed that a real mobile robot could learn to identify the sensor signal that is required for the later action selection and to memorize the signal through reinforcement learning[4]. However, in the task, the robot was required to identify it among five sensor signals, and just to keep it without any processing. To compress the sensor signals and to realize a discrete state representation, an unsupervised learning method was applied other than reinforcement learning. A special RNN architecture was also used.

In this paper, using a general RNN, it is verified that the necessary information can be discovered and memorized from more than one thousand visual signals by the characteristic of being purposive that reinforcement learning has, and finally can be linked to appropriate behaviors.

II. The Marriage of Reinforcement Learning and a Recurrent Neural Network[2]

The system is consisted of one recurrent neural network (RNN) whose inputs are visual signals and whose outputs

Katsunari Shibata is and Hiroki Utsunomiya was with the Department of Electrical and Electronic Engineering, Oita University, 700 Dannoharu, Oita, Japan (email: shibata@oita-u.ac.jp). Hiroki Utsunomiya is now with Kyushu Toshiba Engineering Co., Ltd.

This work was supported by JSPS Grant-in-Aid for Scientific Research #19300070

978-1-4244-9636-5/11/$26.00 ©2011 IEEE

are actuator commands. The RNN is trained by the training signals derived from reinforcement learning algorithm at each time step. When compared with the general approach using reinforcement learning in robotics where the entire system is modularized into some functional modules such as recognition, planning and control, and reinforcement learning is used only for mapping from a state space to an action space, the approach with no given pre-processing may seem to be inefficient at the first impression. However, our approach enables to optimize the entire system by reinforcement learning and supports purposive function emergence based on the optimization. That is expected because the system cannot be optimized to get more reward and less punishment without acquiring necessary functions, and the entire process is optimized consistently and in harmony through the learning of a NN. The approach is also analogous to the fact that in real lives, a nerve system connects from sensors to actuators. For the case of this paper, discovering, recognizing and memorizing arrow meaning is expected. In other words, it is expected to solve a recognition problem in which "what should be recognized" has to be discovered.

Let us see the concrete learning algorithm. Based on reinforcement learning algorithm, training signals are generated, and supervised learning is done. This eliminates the need to supply training signals from outside. In this paper, for a continuous input-output mapping, actor-critic[8] is used as a reinforcement learning method. Therefore, the output of the RNN is divided into a critic output and actor outputs that decide motions. At first, TD-error is represented as

$$\hat{r}_{t-1} = r_t + \gamma P(\mathbf{s}_t) - P(\mathbf{s}_{t-1}) \tag{1}$$

where r_t is the reward given at time t, γ is a discount factor, $\mathbf{s}_t$ is the sensor signal vector at time t, and $P(\mathbf{s}_t)$ is the sum of the critic output and 0.5 when $\mathbf{s}_t$ is the input of the network. Here, since the sigmoid function whose value ranges from -0.5 to 0.5 is used, 0.5 is added to adjust the value range of output function of the NN to the range of the actual critic value. The training signal for the critic output is computed as

$$P_{s,t-1} = P(\mathbf{s}_{t-1}) + \hat{r}_{t-1} = r_t + \gamma P(\mathbf{s}_t), \tag{2}$$

and the training signal for the actor output is computed as

$$\mathbf{a}_{s,t-1} = \mathbf{a}(\mathbf{s}_{t-1}) + \hat{r}_{t-1}\mathbf{rnd}_{t-1} \tag{3}$$

where $\mathbf{a}(\mathbf{s}_{t-1})$ is the actor output when $\mathbf{s}_{t-1}$ is the input of the network, and $\mathbf{rnd}_{t-1}$ is the random number vector that was added to $\mathbf{a}(\mathbf{s}_{t-1})$ as a trial and error factor when the camera moved. Then $P_{s,t-1}$ (actually 0.5 is subtracted to adjust it to the value range of the network output) and $\mathbf{a}_{s,t-1}$ are used as training signals, and the RNN with the input $\mathbf{s}_{t-1}$ is trained once according to BPTT (Error Back Propagation Through Time)[9]. What the readers are requested is to bear in mind that the learning is very simple and general, and as they notice, no special learning for the task is applied. It is focused that the meaning of patterns is discovered and memorized through reinforcement learning with delayed rewards and punishments.

III. Experiment

A. Setups

The environment of the experiment is shown in Fig. 1. Four displays, from Display 0 to Display 3, are allocated on a circle with the radius of 45cm, and at the center of the circle, a movable camera is located. At each episode, one of some blue arrows is chosen randomly, and presented on either of Display 1 or Display 2 that is also chosen randomly. On the next display in the arrow-pointing direction, a red circle is presented, and that indicates the goal state. On the other hand, on the next display but in the opposite direction, a green $\times$-mark is presented, and that indicates a non-goal state. For example, when a right arrow is presented on the Display 2, which is the case of Fig. 1, the red circle is presented on the Display 1, and the green $\times$-mark is presented on the Display 3. Nothing is presented on the Display 0.

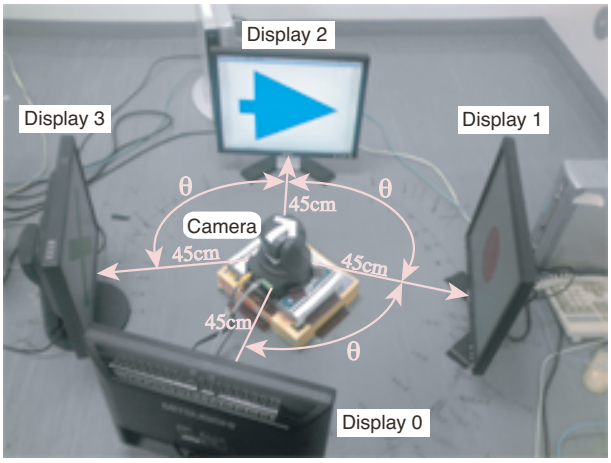


Fig. 1. The environment of the experiment using 4 displays and one movable camera. This display allocation is for the level 7 that is the final level.

At each episode, the camera starts the state facing the presented arrow in its center. The camera moves with the limit of one-dimensional lateral (pan) motion. When the camera catches the red circle, that is, when the red area is large and the center of gravity of it is located within 4 pixels from the image center in the 26×20 camera image, the episode is terminated and a reward is given. When it catches the green $\times$-mark, a penalty is given, but the episode is not terminated immediately. If such state continues for successive 10 time steps, the episode is terminated. Since it is possible that no arrow, no red circle and no green $\times$-mark are shown in the camera image in an episode, the system cannot reach the red circle without memorizing the arrow direction.

At the early stage of learning, the angle between any two neighbor displays is set to 52 degree. This allocation is defined as level 1. In this level, the gap between neighbor displays is very small, and it does not happen that no arrow, no red circle, and no green $\times$-mark are seen. When the system can reach the goal within $(level\_number + 15)$ steps for successive 50 episodes, it moves to the next level. The angle between the displays increases by 8 degrees when

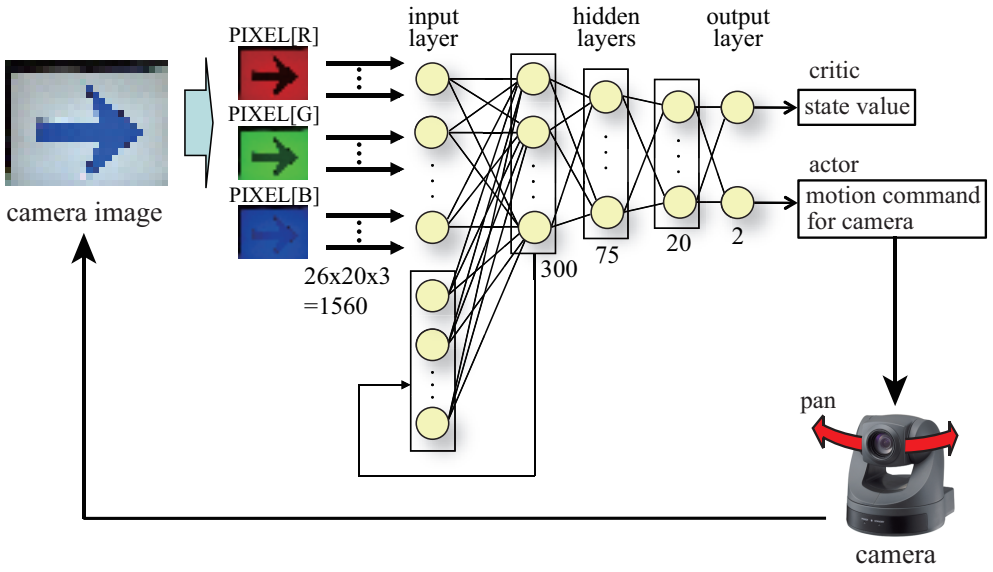
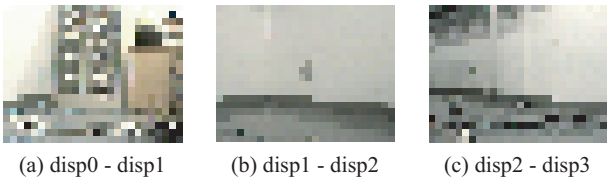


Fig. 3. The architecture of the experimental system and the flow of signals

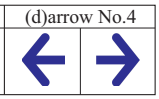


(a) disp0 - disp1 (b) disp1 - disp2 (c) disp2 - disp3

Fig. 2. Sample images between neighbor displays in the case of level 7.

TABLE I

THE ARROWS THAT ARE PRESENTED DURING LEARNING.

(a)arrow No.1	(b)arrow No.2	(c)arrow No.3	(d)arrow No.4
			

moving to the next level. The location of the Display 2 is fixed. The level increases up to the level 7 in which the angle between displays is 100 degrees as shown in Fig. 1.

The reason why four displays are used is as follows. In the preliminary experiment using 3 displays in which an arrow pattern is always presented at the Display 1, the camera did not move according to the memorized arrow direction, but moved according to the difference of the background when the arrow disappeared out of the visual field. If four displays are used, when a left arrow is presented on the Display 1, the same background appears as the case when a right arrow is presented on the Display 2. Therefore, the system cannot behave appropriately only from the background, and has to memorize the arrow direction. For reference, the difference of images between two neighbor displays is shown in Fig. 2 for the case of level 7.

The used camera is EVI-D70 by Sony Co. Ltd., and the used displays are 19inch size with black frame except that only the Display 3 is 17 inch sized. The size of the captured image is 640×480, but due to the limited computational power, it is used after resizing to 26×20 using the OpenCV library. As shown in Table I, 4 patterns for each arrow direction (right or left) are used as arrow images for learning. 7 patterns are used for test as mentioned later. When the distance, which is the sum of the absolute value of the difference in each RGB value between corresponding pixels, from the left arrow No.1 is computed, the right arrow No.1 and No.2 are closest two though the direction is different, and the most distant pattern is the left arrow No.4.

Figure 3 shows the system architecture and signal flow. In this system, since the formation of memory through learning is required, a RNN is used as mentioned. Considering the many input signals and expecting the formation of abstract representation, a five-layer neural network is employed. The outputs of the lowest hidden layer that is the closest to the input are fed-back to itself at the next time step. The camera image that is shrunk to $26 \times 20 = 520$ pixels of RGB values are directly put into the RNN as inputs after the linear transformation between -0.5 to 0.5. The total number of input signals is $26 \times 20 \times 3 = 1,560$. Since only pan motion is used in the camera, there is only one actor output. According to the actor output after adding a random number rnd_t, the camera moves laterally.

The numbers of neurons in the hidden layers are 300, 75, 20 from the lowest. The initial weights for self-feedback connections are set 4.0, while those for the other feedback connections are set 0.0 for efficient and non-divergent error propagation to the past states and for easy formation of memory. The other initial connection weights are set randomly between -1.0 to 1.0. The number of outputs is two; one is for critic and the other is for actor. As a non-linear output function, sigmoid function whose range is from -0.5 to 0.5 is used in each neuron in the hidden and output

layers. The training signals are limited from -0.4 to 0.4 to avoid the output from being in the saturation area of the sigmoid function. The camera is rotated as the angle that is proportional to the actor output (a random number rnd_t is added as a trial and error factor in learning phase), and it rotates about 7.7 degree when the actor output is 0.4. The output for critic is used actually after adding 0.5. The reward at the goal state is 0.9, and the training signal for critic is $0.9 - 0.5 = 0.4$ from Eq. (2) with $P(\mathbf{s}_t) = 0$. When the camera moves to the opposite direction to the arrow pointing, and catches the green $\times$-mark around the center of the image, -0.1 is substituted for r in Eq. (2) as a penalty, and the episode continues. However, if the state continues for 10 time steps, the episode is terminated. Otherwise, the neural network is trained according to the training signal as Eq. (2) with $r_t = 0$. The trace-back steps in BPTT is truncated at 10 steps. The random number rnd_t that is added for trial and errors is a uniform random number whose range is decided according to

$$rnd\_range = exp(-goal\_count * 0.001)/2 \qquad (4)$$

where $goal\_count$ increases by 1 when the system reaches the goal state and decreases by 4 when it does not reach the goal state. If $goal\_count$ is set to 0 when it becomes below 0. By this setting, the range of the random number decreases when the system continues to succeed, and increases largely when it fails. The maximum number of steps in one episode is set to 40. When the system does not reach the goal state in the number of steps, the episode is terminated and is counted as a failure. The learning rate is 0.125 for the feedback connection weights and 0.5 for the others. The constant input for bias is 0.1, and each bias is trained as a connection weight.

B. Results

Fig. 4 shows the learning curve for each display-arrow_direction pair. The vertical axis indicates the number of steps to the goal state at each episode. All the levels of learning finished after 2,780 episodes. In the level 1 learning, it took 1,089 episodes. Around the 300th episode, the camera always went to the right direction not depending on the presented arrow direction, while around a little before the 500th episode, it always went to the left direction. However, after the level 2, it did not happen that the camera continued to catch the $\times$-mark on the center except for the case of right arrow on the display 1 in the level 4. In the level 4 learning, it took more than 500 episodes, but in the other levels, it took less than 500 episodes. Since the angle between displays increased as the level goes up, the minimum number of steps to the goal state becomes larger gradually. The difference between the presented displays does not seem so much, but a small difference can be seen in the level 4.

Fig. 5 and Fig. 6 show two sample camera motions after learning. Fig. 5 shows the case when the left arrow No.1 was presented on the Display 1, and Fig. 6 shows the case when the right arrow No.4 was presented on the Display 2.

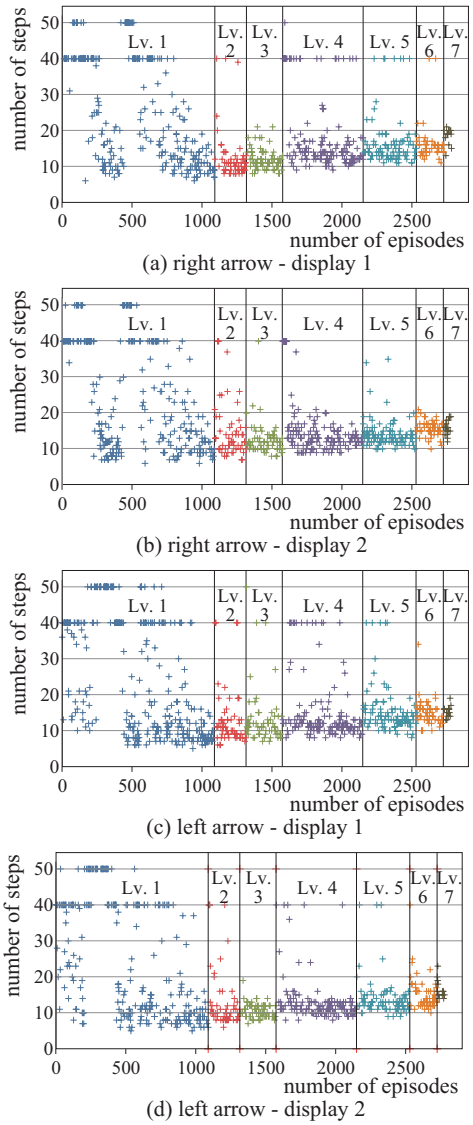


Fig. 4. Learning curve for each display-arrow_direction pair. The plot at 50 steps indicates that the camera caught the green $\times$-mark for successive 10 times, and the episode was terminated. The plot at 40 steps indicates that the camera could not reach the goal state within 40 steps and the episode was terminated.

In each test episode, no random numbers were added to the actor output when the camera motion is decided. In each figure, the camera image at each step and the change of critic and actor output are shown. In the both figures, the critic increases almost monotonically and is close to the ideal curve. In the case of left arrow, the actor output is negative during the episode, and on the other hand, in the case of right arrow, the actor output is positive. In the middle of each episode, which is around the step 7, 8 or 9, the arrow pattern disappears, and the image is almost the same between the two cases because in the both cases, the camera passes the same situation as can be seen in Fig. 1. Nevertheless, the critic increases monotonically, and the actor output is completely different between the two cases even though the input image looks very similar. This means that the system discovered

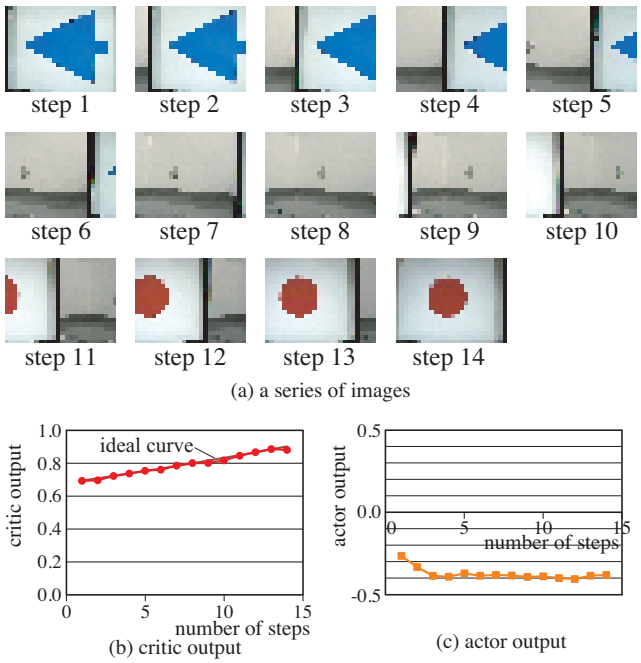


Fig. 5. The change of captured image by the camera motion and the change of the critic and actor outputs during one episode when the left arrow No.1 is presented on the Display 1.

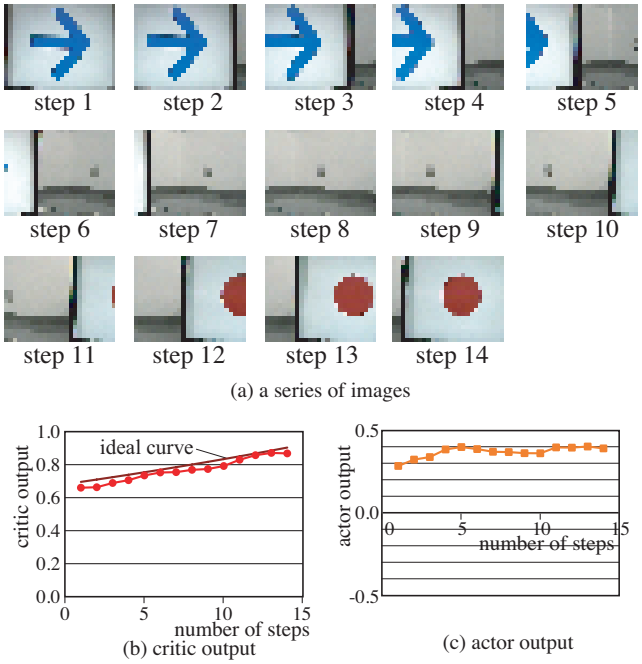


Fig. 6. The change of captured image by the camera motion and the change of the critic and actor outputs during one episode when the right arrow No.4 is presented on the Display 2.

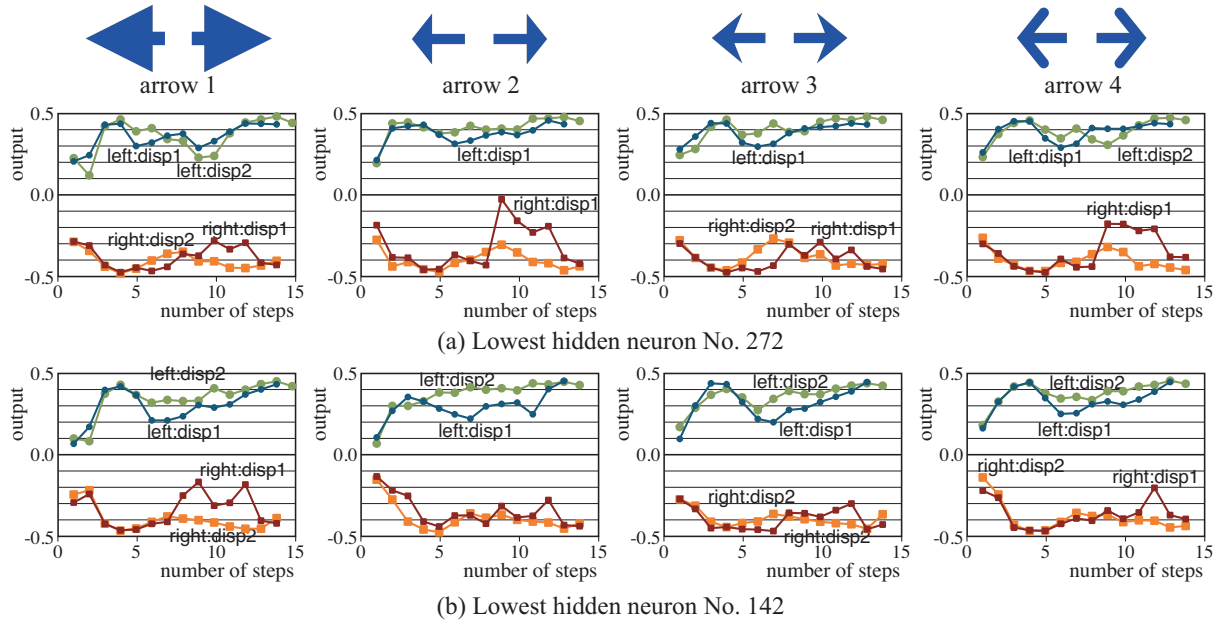


Fig. 7. The output change of two lowest hidden neurons during one episode for each learning arrow pattern. They seems to contribute to memorizing the arrow direction "left:disp1" indicates left arrow presented on the Display 1.

that the arrow direction is important among more than one thousand visual signals, and learned to extract and memorize the arrow direction. Then the appropriate state evaluation and behavior were achieved using the memorized information.

Next, the output change during one episode for 4 learning arrow patterns are observed in all the 300 lowest hidden neurons after learning, and two neurons that look to contribute strongly to extracting and memorizing the arrow direction are picked up. The output changes are shown in Fig. 7. The graph is drawn for each of the 4 learning arrow patterns. In these neurons, it seems that not depending on the arrow patterns, the output is positive when the arrow direction is left, and the output is negative when the arrow direction is right even after the arrow disappeared. In the case of right arrow on the Display 1, the output sometimes goes close to the 0.0. It may be due to the difference in image between the

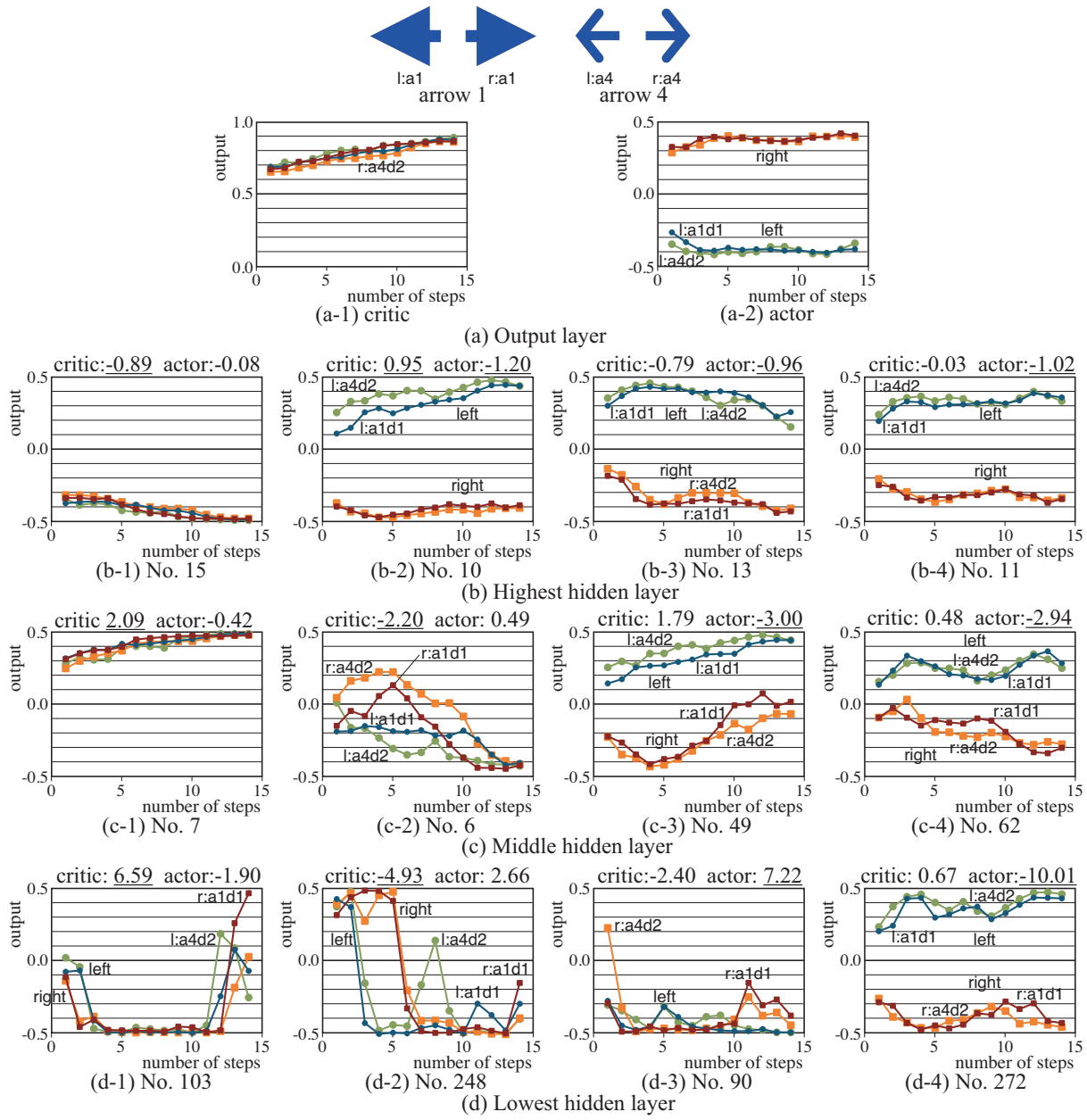
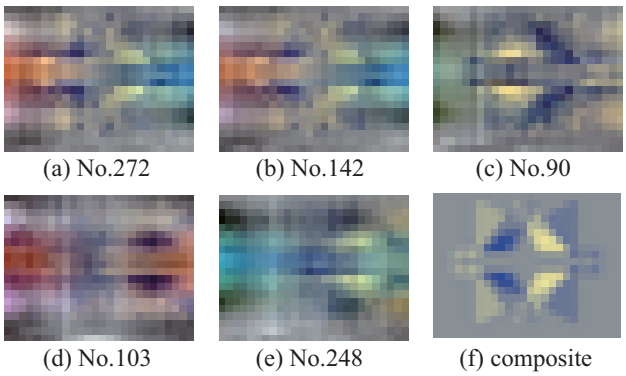


Fig. 8. The output change of some neurons during one episode for the four cases of right and left arrow No.1 on the Display 1 and the right and left arrow No.4 on the Display 2. The hidden neurons seem to contribute much to critic or actor by judging from the connection weights. Two values on each graph indicate the contribution to the critic and actor on the assumption that the transformation in each neuron is linear. The graphs in the left-half are for the neurons that seem to contribute the critic, and on the other hand, the graphs in the right-half seem to contribute to the actor. No. 10 neuron in the highest hidden layer (b-2) seems to contribute to both critic and actor. "l:a4d1" indicates the case when the left arrow No. 4 is presented on the Display 1.

displays as shown in Fig. 2. It is also seen that the difference in output between the arrow directions is not so large at the initial state, but increases in a couple of steps. That suggests that an associative memory is formed in the RNN. Actually, the connection weights between these two neurons are small, but positive though they were 0.0 initially.

By assuming the transformation of each neuron is linear, the contribution of each hidden neuron to the critic or actor can be represented as a value. Fig. 8 shows the output change of the neurons that seem to contribute the most or the second most to the critic or actor in each hidden layer together with the critic and actor output changes. The highest hidden neuron No. 10 contributes the most to both the critic and actor, and the case of No. 13 neuron whose contribution to the actor is the third most is also shown. As for the lowest hidden layer, since the output of the neuron that contributes the second most to the critic does not change so much during one episode, the neuron No. 248 that contributes the third-most to the critic is shown. The lowest hidden neuron No. 142 that was shown in Fig. 7 is the third-most to the actor, and the output change is almost the same as the No. 272.

The output change of the neurons in the highest hidden



(a) No.272 (b) No.142 (c) No.90

(d) No.103 (e) No.248 (f) composite

Fig. 9. The change of connection weights to each of some hidden neurons during learning is visualized as a color image by the linear transforming the change of each weight to the range from 0 to 255.

layer is smooth and the close to the critic or actor even though No. 10 and No. 13 contribute to the both. On the other hand, in the lowest hidden layer, The output change in No. 142 or No. 272 neurons is similar to the actor, but no neurons that increase or decrease smoothly like critic are found.

Fig. 9 shows the change of connection weights through learning in the lowest hidden neurons mentioned above. In each hidden neuron, the weight changes through learning are linearly transformed to the range from 0 to 255 at first. Since the number of connection weights is equal to the number of pixels of the input image, the weight changes in one lowest hidden neuron can be visualized as a color image whose size is the same as the input image. The linear transformation from each weight value to the corresponding pixel value $f_{c,i,j}$ that ranges from 0 to 255 is as

$$f_{c,i,j} = (int)\left(\frac{Wa_{c,i,j} - Wb_{c,i,j}}{\max_{c,i,j}|Wa_{c,i,j} - Wb_{c,i,j}|} \times 127\right) + 128, \quad (5)$$

where Wa, Wb indicate the weight after and before RL respectively, c indicates the color and can be R, G, or B, and i, j indicate the raw and column number of a pixel in the image respectively. By this transformation, if the value of a connection weight increases through learning, the color of the pixel corresponding to it becomes bright, and if the value decreased, the color of the pixel becomes dark.

The first three neurons contribute much to the actor, and the next two contribute mainly to the critic. The first two images, (a) and (b), are very similar, and the left-half is almost symmetrical to the right-half, but the color is inverted. The color yellow is the opposite color of blue. The system seems to extract the arrow direction from the image at the initial state in each episode. At the pixels that are blue for all the four right arrow patterns and are white for all the four left arrow patterns, which are around the head of the right arrow, the color is strong yellow, and around the head of the left arrow, the color is strong blue. That is a reasonable to detect the arrow direction. Furthermore, yellow right arrow No. 1 and blue left arrow No. 1 are ambiguously seen. Fig. 9 (f) shows the composite image of average over four left arrow patterns and the color-inverted image of the average over four right arrow patterns. It is similar to (a) or (b). The composite of an arrow pattern and the color-inverted image of the same arrow with the other direction is thought to be helpful to eliminate the effect of the lighting condition. It is also known that the neuron detects the arrow direction considering many pixels in parallel. One more thing we can notice is that the left part of the image is reddish and the right part is cyanic that is the opposite color of red. That is thought to discriminate whether the red circle appears on left side or the right side of the camera image, and to contribute to moving left when the red circle appears on the left side, and moving right when it appears on the right side. The red circle comes from the left side when a left arrow is presented and the camera moves to the left. It is thought that to realize the same actor output, the neuron got to recognize both the arrow direction and the place where the red circle appeared.

From the image of the neuron No. 90 (c), it seems to detect the right arrow No. 3 or No.4, that matches the graph in Fig. 8(d-3). As for the neuron No. 103 and 248, which contributes much to the critic, it seems difficult to give appropriate explanation. Anyway, since there are many other neurons processing in parallel, and one neuron makes multiple roles in parallel like the neuron 272 or 142. it is impossible to understand the entire process of the neural network. It is similar to the situation that we try to understand our brain exactly that is a massively parallel and flexible system.

Finally, generalization ability is examined by checking the performance for the test arrow patterns that are not presented in learning. Table II shows the number of steps to the goal state for each arrow pattern for the presentation on each of the Display 1 and Display 2. Fig. 10 shows the critic, actor and the lowest hidden neuron No.272, which is considered to memorize the arrow direction, for all the combinations of arrow pattern, direction and presented display including the learning patterns. In all the patterns except for the case of right arrow No. 9 on the Display 1 (r:a9d1), the camera reached the correct goal even though the number of steps is sometimes more than 20 for the case of right arrow. Except for the failure case, the actor value is positive for the right arrows, while it is negative for the left arrows.

The lowest hidden neuron No.272 also discriminates the arrow direction and keeps it even after the arrow pattern disappeared except for the failure case. In the cases in which more steps than the others are necessary to reach the goal state, which are r:a5d2, r:a6d1, r:a6d2, the neuron takes a value around 0.0 at first. Therefore the actor value is not so large, and so it takes more steps to the goal. However, the neuron is entrained gradually, and finally it converges around a large negative value. In the failure case, the neuron output converged to the incorrect value at first and then kept it for a while. The camera motion for the failure case is shown in Fig. 11. The camera moved to the left (incorrect) direction until it caught a part of the green $\times$-mark. When it caught the green $\times$-mark at around the 13th step, the hidden neuron

TABLE II

THE NUMBER OF STEPS TO THE GOAL FOR EACH ARROW_PATTERN-DIRECTION PAIR. THE TWO NUMBERS INDICATE FOR THE CASES OF PRESENTATION ON THE DISPLAY 1 (LEFT) AND DISPLAY 2 (RIGHT). × INDICATES THAT THE SYSTEM FAILED TO REACH THE GOAL. THE ARROW PATTERNS FROM NO.5 TO NO.11 WERE NOT USED IN LEARNING.

(a)arrow No.1		(b)arrow No.2		(c)arrow No.3		(d)arrow No.4	
14	15	14	14	13	14	14	14	13	14	14	14	13	14	14	14
(e)arrow No.5		(f)arrow No.6		(g)arrow No.7		(h)arrow No.8	
14	16	16	19	14	15	26	20	14	15	16	17	14	16	15	16
(i)arrow No.9		(j)arrow No.10		(k)arrow No.11			
14	15	×	16	13	14	14	14	14	16	14	15				

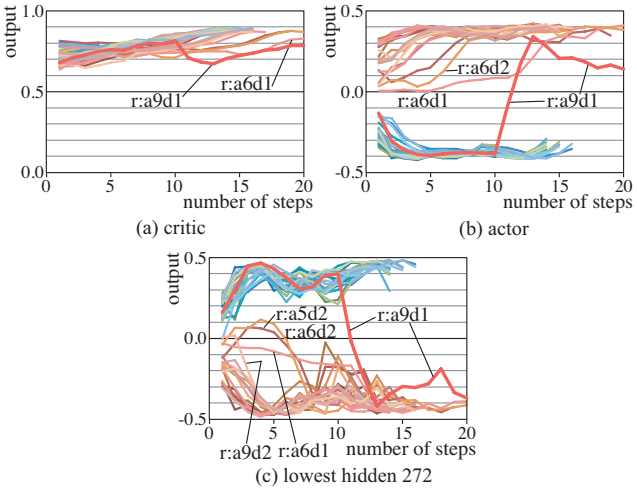


Fig. 10. The change of critic, actor and the lowest hidden neuron No. 272 during one episode for all the combinations of arrow directions, arrow patterns and presented displays. The system failed only for the case of right arrow No. 9 on the Display 1 (r:a9d1, red thick line).

output suddenly went down to the large negative value, and actually it moved to the right direction. However, the actor is not so large, and when it caught the tail of the arrow pattern at around the 25th step, it went to the left direction again. The reason of the failure might be that in the case of the pattern No.9, the distance to the average of four learning patterns of the same direction is not so different from the distance to the average of four learning patterns of the opposite direction, and furthermore, and as can be seen in Fig. 11, the initial position of the camera was slightly biased to the left direction unintentionally.

IV. CONCLUSIONS

Only through reinforcement learning with a reward at a goal state and a penalty at a non-goal state, a system with a real movable camera and a recurrent neural network

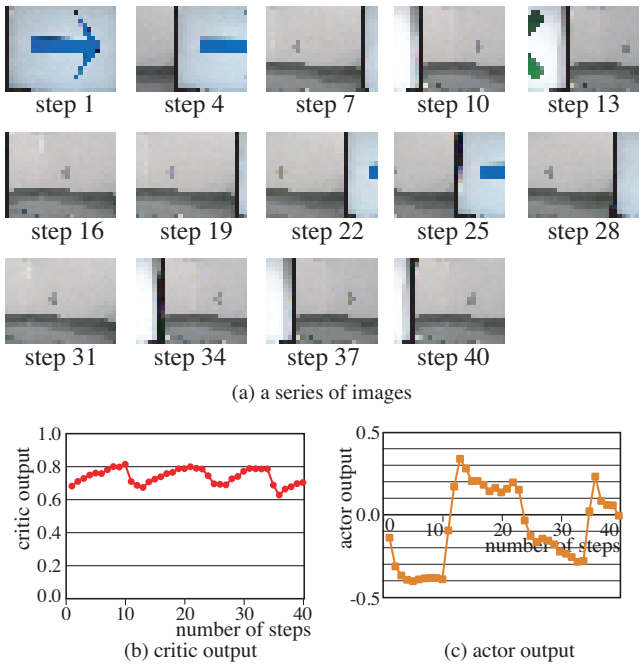


Fig. 11. The change of captured image by the camera motion and the change of the critic and actor outputs when the right arrow No.9 is presented on the Display 1.

discovered that the arrow direction is important, and acquired the recognition of the direction from 1,560 visual signals. The system also got to memorize the recognized arrow direction and to move the camera to the correct direction even after the arrow disappeared out of the visual field.

Introducing a small random shift of the initial arrow location made the learning difficult. Furthermore, the way of exploration seems inefficient. Some improvements are required.

REFERENCES

[1] R.A. Brooks, "Intelligence without Representation," *Artificial Intelligence*, vol. 47, pp. 139-159, 1991.

[2] K. Shibata, "Emergence of Intelligence Through Reinforcement Learning with a Neural Network," *Advances in Reinforcement Learning* (Open access) Intech Open Access Publisher, 2011.

[3] B. Bakker, F. Linaker, J. Schmidhuber, "Reinforcement Learning in Partially Observable Mobile Robot Domains Using Unsupervised Event Extraction", *Proc. of IROS 2002*, Vol. 1, pp. 938?943, 2002

[4] B. Bakker, et al., "A Robot that Reinforcement-Learns to Identify and Memorize Important Previous Observations," *Proc. of IROS 2003*, pp. 430-435, 2003.

[5] A. Onat, H. Kita, and Y. Nishikawa, Q-Learning with Recurrent Neural Networks as a Controller for the Inverted Pendulum Problem. Proc. of ICONIP 98, 837-840 (1998)

[6] Utsunomiya, H. & Shibata, K. (2009). Contextual Behavior and Internal Representations Acquired by Reinforcement Learning with a Recurrent Neural Network in a Continuous State and Action Space Task, *Advances in Neuro-Information Processing*, LNCS, Vol. 5506, pp. 755-762

[7] Goto, K. & Shibata, K. (2010). Emergence of Prediction by Reinforcement Learning Using a Recurrent Neural Network, *J. of Robotics*, Vol. 2010, Article ID 437654

[8] A.G. Barto, et al., Neuronlike Adaptive Elements That can Solve Difficult Learning Control Problems. *IEEE Trans. of SMC*, Vol. 13, pp.835-846, 1983.

[9] D.E. Rumelhart, et al., "Learning Internal Representation by Error Propagation," *in Parallel Distributed Processing*, 1986.