Autonomous Reusing Policy Selection using Spreading Activation Model in Deep Reinforcement Learning

Yusaku Takakuwa[1]
Department of Information and Communication Engineering
Tokyo Denki University
Tokyo, Japan

Hitoshi Kono[2]
Department of Engineering
Tokyo Polytechnic University
Kanagawa, Japan

Hiromitsu Fujii[3]
Department of Advanced Robotics
Chiba Institute of Technology
Chiba, Japan

Wen Wen[4]
Department of Precision Engineering
The University of Tokyo
Tokyo, Japan

Tsuyoshi Suzuki[5]
Department of Information and Communication Engineering
Tokyo Denki University
Tokyo, Japan

***Abstract*—This paper describes a policy transfer method of a reinforcement learning agent based on the spreading activation model of cognitive psychology. This method has a prospect of increasing the possibility of policy reuse, adapting to multiple tasks, and assessing agent mechanism differences. In the existing methods, policies are evaluated and manually selected depending on the target–task. The proposed method generates a policy network that calculates the relevance between policies in order to select and transfer a specific policy that is presumed to be effective based on the current situation of the agent while learning. Using a policy network graph structure, the proposed method decides the most effective policy while repeating probabilistic selection, activation, and spread processing. In the experiment section, this study describes experiments conducted to evaluate usefulness, conditions of use, and the usable range of the proposed method. Tests using CartPole and MountainCar, which are classical reinforcement learning tasks, are described and transfer learning is compared between the proposed method and a Deep Q–Network without transfer. As the experimental results, usefulness was suggested in the transfer learning of the same task without manual compared with previous method with various conditions.**

***Keywords*—Reinforcement learning; transfer learning; deep learning; cognitive psychology; spreading activation theory**

I. Introduction

In recent years, practical realization of robots, which can perform flexibly in various environments like those of a human being, is being expected. It is difficult for robots to act flexibly in unknown and dynamic environments without human control. In addition, it is also difficult to establish a control strategy beforehand. Therefore, many researches that allow the robot to learn by itself are being actively conducted [1] [2]. In these researches, reinforcement learning is used for robot autonomous learning. Reinforcement learning is a method that allows an agent (hereinafter, a learning robot or system will be referred to as an agent) to learn the optimal action by trial and error [3]. Reinforcement learning enables agents to autonomously acquire behavior rules, however there are still some problems that need to be addressed, such as the extensive learning time required for practical and complex tasks. In order to solve this problem, a learning method of reusing a policy of a previously learned task (source–task) for a new task to be learned (target–task) is proposed. This learning method, called transfer learning [4], has been studied in various domains. This method enables agents to improve the adaptability of the target–tasks, thus shortening the learning time. However, most of the transfer learning in the existing research needs to previously determine the policy of the source–task that is considered to be effective for the target–task. Therefore, it is necessary to manually evaluate the policies.

In the existing methods, selecting and reusing effective policies from a plurality of them has been proposed [5] [6]. However, it is considered difficult adapting to obstacles when reusing policies. Specifically, many obstacles should be considered, such as performing different tasks (heterogeneous tasks), utilizing different agent mechanisms (heterogeneous agents), and setting the parameters of reinforcement learning itself, among others. Assuming that agents are used in various fields and environments, it is very difficult for people to set up reinforcement learning parameters considering all obstacles. Therefore, this research discusses a new policy reuse method with a prospect to reduce the manual labor required for policy selection and that can perform flexibly in the presence of various obstacles.

For the above problems, in order to flexibly select a policy,

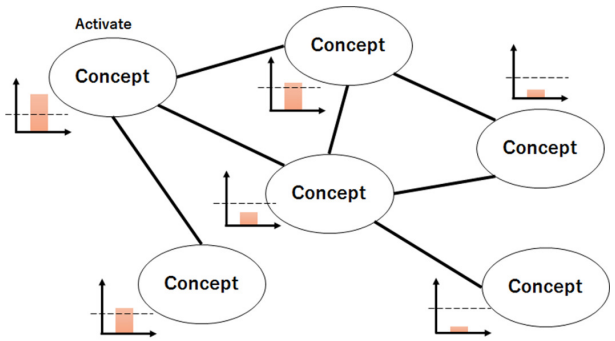

Fig. 1. Simplified Image of the Spreading Activation Model. A Value Called Activation Value Diffuses Through the Network Extending from the Activated Concept.

we apply the spreading activation model as a psychology model that aids human behavior and judgment in the proposed policy selection method based on previous method [7]. In addition, Deep Q-Network method is adopted as function approximation of policy of reinforcement learning, and proposed method is evaluated with various classical reinforcement learning tasks in computer simulation.

A. Spreading Activation and Existing Research

The spreading activation model [8] is a psychology model related to human concept formation (remembrance, re-recognition, etc.) on the assumption that concepts acquired by humans are stored as a network structure in the brain. Many concepts are memorized using schematic representation as network structure in human brain. The concept has an activation value that can be activated or deactivated by external stimuli such as visual information. In the process, activation value increases beyond the threshold value, the concept remain. This phenomena is called recall. Activated values spread to related concepts that are connected via path of semantic distance. In the spreading activation model, there is a concept called semantic distance in which the distance between concepts varies according to the strength of the relevance between these concepts. Concept activation is done via a relevance network. An example of an activation diffusion model is shown in Fig. 1.

Kono *et al.* are proposed transfer reinforcement learning with spreading activation model which is called SAP–net to select the policies adaptivelly according to environments [7]. SAP–net is discussed effectiveness by simplified computer simulation such as shortest path problem, and it is defined theoretically for implementation. However SAP–net is not consider the computer resources for actual implementation in the robot or agent, which means that it is adopted table type Q function to be described a policy.

B. Policy Description Method

When reusing policies acquired through reinforcement learning, it is important to describe and store them. In reinforcement learning, policy description methods can be roughly divided into two types. One is to describe the behavioral value obtained by learning in a Q function as Q–table which is constructed by look–up table. It is indirectly prescripts the policy. In the case of using the Q–table, learning is possible by describing all the state-behavior pairs of the agent. By mapping the behavior value from the Q–table for each state, it is possible to perform the transfer learning. However, when using a Q–table, the table size increases exponentially as the number of states increases, which is disadvantageous. Therefore, it is difficult to learn tasks handling many state numbers like in real environments and complicated tasks.

As a second policy description method, there is a method to approximate the behavior value by a function. By function approximation, it is possible to learn tasks with a larger number of states than when using Q–tables. Various methods such as tile coding [9] [10], fuzzy [11] [12], RBFN [13] [14], and deep neural networks [15] have been proposed as function approximation methods. In this research, we discuss a function approximation method using neural networks (NN), which have been actively researched recently, such as Deep Q–Networks, Deep Reinforcement Learning [16] [17] [18] and so on. By function approximation, it is possible to learn tasks with more states than when using a Q–table. However, depending on the setting of the intermediate layer of a NN and the activation function when performing function approximation, it may cause an increase in learning time, excessive learning, and unlearning. In the case of a task performed by an agent in a real environment, it has been considered that transfer learning using a function approximation method is effective; however, it is not realistic to optimize the network structure of a NN for each learning environment, scale, and task.

C. The Principal Aim of this Study

From the above context, the principal aim of this study is to discuss the following two topics.

1) The proposed method of automatically selecting effective policies from the policy features, such as the setting of the hidden layers of a NN.
2) The prospects of the proposed method, conditions of use, and applicable range.

As a function approximation with reinforcement learning method, Deep Q–Network is implemented in this study. Therefore proposed method is based on policy selection method in transfer reinforcement learning with spreading activation model proposed by Kono *et al.* [7], and is tuned that the policy selection method can be implemented by Deep Q–Network as a part of reinforcement learning algorithm. In the experiment, CartPole and MountainCar are adopted as evaluation function, which are classical reinforcement learning tasks.

II. Methods

A. Precondition

In this section, to introduce the proposed method, we describe related terms and assumptions. For the reinforcement learning algorithm, Q learning was used.

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\{r + \gamma \max Q(s',a)\} \quad (1)$$

Here, observable state is $s \in S$, and action of agent is $a \in A$. S and A are assumed discrete state. Learning parameters

are learning rate α, discount rate γ and reward$r$. In addition, hereinafter, the NN model used for function approximation is called policy.

B. Flow of the Proposed Method

Initially, the proposed method classifies each policy into categories based on features of multiple policies learned in advance, and creates a network using these categories. The proposed method selects a policy from the selection probability calculated based on a parameter, which is called the activation value, given to each policy. Policies are selected by referring to selection probabilities. Based on the loss between the value calculated for each action obtained through the policy and the value that acts as the teacher, the action obtained by the selected policy is classified as one that promotes learning (positive transition) or one that does not promote learning (negative transition). From this judgment result, the network is constructed based on the selected policies. By performing this process for each action, the activation value of each policy is changed during the transition learning. Thus, a system that assigns preferential learning to policies with large activation values is constructed.

C. Categorize Policies

Based on the features of the policies, multiple policies are categorized. A category in this research refers to a set of relevant policies that are calculated from the relevance of multiple policies. Policies are evaluated from one viewpoint and judged whether they belong to the same category. This viewpoint is called a prototype. In this study, the prototype is determined empirically based on the information available from the features included in the learned policies. For each category, a policy distance d_{ij}, which describes the relationship between the policies, is generated. The inter-policy distance combines all the policies to cover all the connection patterns in the category.

Multiple benefits can be obtained by classifying policies into categories. First, it is possible to summarize and manage similar policies using a NN of multiple learned policies. It is thought that policies having similar structures of NN layer number and unit number are likely to obtain similar learning results when the other parameters are unified.

Second, we can classify policies learned by with heterogeneous tasks or heterogeneous agents. Policies learned by different tasks and agent mechanisms are difficult to learn with different learning settings. Therefore, categories are useful for organizing and managing policies.

D. Policy Network (SAP–Net)

A policy network is created using the classified categories. The generated policy network is called SAP–Net in this research. SAP–Net is an undirected graph, and it is defined as follows.

$$\mathbb{G} = (\Pi, E, \omega) \tag{2}$$

where Π is the set of policies to be combined, E shows the path connection relation between policies, and ω is the weight of the distance between policies. The origin of the set of policies, which is the vertex of the undirected graph, is $\pi_i \in \Pi, e \in E\ (e = \pi_i, \pi_j)$. The network structure of the graph is expressed using an adjacency matrix. The adjacency matrix is represented by a square matrix, and defines the elements in the matrix as follows: Fig. 2 shows an example of SAP-Net. A policy has able to multiple network path as output and input as shown in Fig. 3.

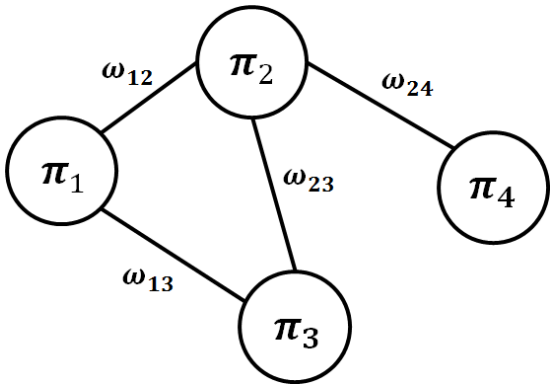

Fig. 2. SAP-Net. Multiple Policies are Tied by Policy Distance d_{ij}.

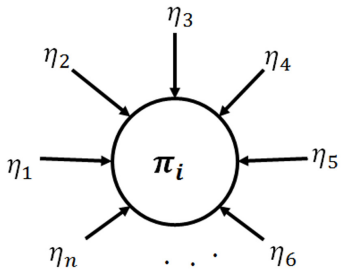

Fig. 3. Activation Value Inputs and Outputs η Flow into the Policy π_i.

$$M_{ij} = \begin{cases} 0 & (e \notin E) \\ \omega_{ij} = 1 & (e \in E) \end{cases} \tag{3}$$

Shown in Fig. 2 by an adjacency matrix, the networks can be expressed as Eqn. (4). π_i shows the Qtable and an approximate model, which are the policies already acquired by the source–task

$$M = \begin{pmatrix} 0 & \omega_{12} & \omega_{13} & 0 \\ \omega_{12} & 0 & \omega_{23} & \omega_{24} \\ \omega_{13} & \omega_{23} & 0 & 0 \\ 0 & \omega_{24} & 0 & 0 \end{pmatrix} \tag{4}$$

When constructing SAP-Net, weights of each element are adjusted by influencing the weight of inter-policy distance in the prototype matrix generated from the classified categories. The prototype matrix is represented by an adjacency matrix like SAP-Net, and weight ω_c is generated from the connection between the policy distances in a given category, as shown in Eqn. (5).

$$M = M + \delta(\mathbb{P}_1 + \cdots + \mathbb{P}_n) \tag{5}$$

n is the number of categories, and δ is a coefficient for adjusting the weight of the prototype matrix.

To calculate similarities and relevance of the policies, it is necessary to evaluate them from various viewpoints. In general, when making a judgment that A and B are related or similar, we count the number of related parts and consider that they are more similar as the number of related parts increases. In this research, we apply this idea and calculate relevance by adding up the prototype matrices generated with various prototypes.

E. Selection of Policy

To select the policy, the selection probability $P(\pi_i)$ of each policy is calculated based on the constructed SAP-Net and the activation value $\mathbb{A}_i$ given to each policy. The calculation formula is shown in the following Eqn. (6). i, j are the numbers used to identify policies.

$$P(\pi_i) = \frac{\exp(\mathbb{A}_i)}{\sum_{j=0}^{n-1} \exp(\mathbb{A}_j)} \tag{6}$$

F. Activation Process

In this study, based on the loss $\mathbb{E}_t^{\pi_i}$calculated when the agent acts by selecting a certain policy π, judgment of positive and negative transitions is made. The judgment formula is shown in Eqn. (7).

$$T_a = \mathbb{E}_{t+1}^{\pi_i} - \mathbb{E}_t^{\pi_i} \tag{7}$$

Here, T_a is threshold value for judgment. Based on the calculated judgment result, increase processing is applied to the activation value of the policy. The process applied to the activation value is shown in Eqn. (10). $\mathcal{A}$ is a activated coefficient that adjusts the activation value increment. This process is called activation, and the activation value must be adjusted.

$$\mathbb{A}_i = \begin{cases} \mathbb{A}_i + \mathcal{A} & (T_a \leq 0) \\ \mathbb{A}_i & (T_a > 0) \end{cases} \tag{8}$$

G. Spreading Process

The spreading process of the activation value is applied to a policy with connection relation via policy distance extending from the activated policy. When defining the spreading value, the state of activation value η diffuses to the policy π stored on SAP-Net, as shown in Fig. 4. The activation value output is recorded when a certain policy is activated or diffused. The activation value output η_i from a certain policy π_i is obtained by the following expression using the number of policy distances extending from the policy π.

$$\eta_i \leftarrow \frac{\mathcal{A}}{k} \tag{9}$$

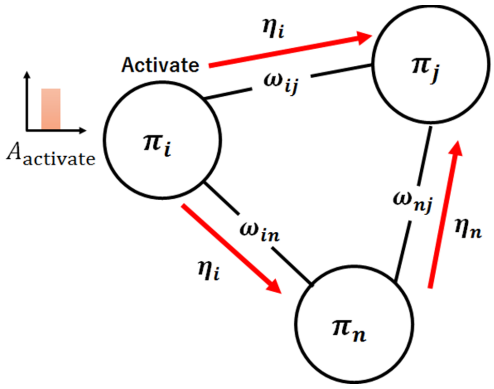

Fig. 4. The Activated Value Outputs η Propagate from the Activated Policy π_i.

Due to diffusion, the activation value output is updated every time via a policy. This is done based on Eqn. (9). Considering the diffusion of the activation value for the two sources of diffusion π_i and the diffusion destination π_j adjacent to the diffusion source, the active value change amount $\Delta\mathbb{A}_j$ takes into account the activation value output η_i from π_i and the diffusion through multiple policies. The sum $\sum\omega$ of the weights of the inter-policy distances by which it passed through, and h of inter-policy distances by which the intermediary passed through are obtained as follows.

$$\Delta\mathbb{A}_j = \begin{cases} 0 & (\sum\omega \geq T_\omega) \\ \dfrac{\eta_i}{h\sum\omega} & (\sum\omega < T_\omega) \end{cases} \tag{10}$$

An image of activation diffusion is shown in Fig. 4. Activation spreading diffuses permanently as long as it does not specify the range of spreading, because it is calculated recursively through each policy. Taking this into consideration, the threshold value T_ω is determined by the sum of the weights of the inter-policy distances by which the value passes through. Ultimately, each policy calculates the total sum of activation change amounts, as shown in Eqn. (11), by the number of active value outputs received through the network and adds it to the residual activation value. In the proposed method, we adjust the activation value of each policy through these processes, and re-learn while choosing a policy for each behavior. After the learning, the policy with the highest activation value is determined as a policy that can be learned effectively. The two algorithms describing the activation process and the spreading process of the proposed method are shown in Algorithm 1, 2.

$$\mathbb{A}_j = \begin{cases} \mathbb{A}_j + \sum\Delta\mathbb{A}_j & (T_a \leq 0) \\ \mathbb{A}_j & (T_a > 0) \end{cases} \tag{11}$$

Each policies are labelled environmental information of source task respectively. Therefore input of proposed system is environmental information of target task, and final output of proposed system is selected policy or policies.

Algorithm 1 Activation process

function ACTIVATION($i, \mathbb{A}, M, \mathcal{A}$)
  Initialize: $h = k = j = \Omega = 0, N$ is set as number of policies collected via source–task
    ▷ h is inter–policy distances by witch the intermediary passed through ▷ k is number of path that π_i has
    ▷ j is number of policy connected to π_i via d_{ij} ▷ Ω is corresponding to $\sum \omega$
  $\mathbb{A}_i \leftarrow \mathbb{A}_i + \mathcal{A}$
  for $j = 0$ to $N - 1$ **do**
    if $M_{ij} \neq 0$ **then**
      $k \leftarrow k + 1$
    end if
  end for
  $j = 0$
  while $j < N$ **do**
    if $M_{ij} \neq 0$ **then**
      $\eta_i = \frac{\mathcal{A}}{k}$
      $\Omega \leftarrow \Omega + M_{ij}$
      SPREADING(j, h, Ω, η_i)
      $\Omega \leftarrow \Omega - M_{ij}$
    end if
    $j \leftarrow j + 1$
  end while
end function

Algorithm 2 Spreading process

function ACTIVATION(j, h, Ω, η_i)
  Initialize: $k = 0$
  $\eta = \eta_i$
  $i = j$
  $h \leftarrow h + 1$
  if $\Omega < T_\omega$ **then**
    $\mathbb{A}_i \leftarrow \mathbb{A}_i + \frac{\eta}{h\Omega}$
    for $i = 0$ to $N - 1$ **do**
      if $M_{ij} \neq 0$ **then**
        $k \leftarrow k + 1$
      end if
    end for
    $j = 0$
    while $j < N$ **do**
      if $M_{ij} \neq 0$ **then**
        $\eta \leftarrow \frac{\eta}{k}$
        $\Omega \leftarrow \Omega + M_{ij}$
        SPREADING($j, h, \sum \omega, \eta$)
        $\Omega \leftarrow \Omega - M_{ij}$
      end if
    end while
  end if
end function

III. EVALUATION

A. Experimental Setup

This study conducted transition-based learning using the proposed method by establishing multiple approximate models that randomly set and learned hidden layers to verify the usefulness of it. The learning efficiency of reinforcement learning using this new transfer learning and normal Deep Q–Network was compared. In this experiment, we focused on the structure of the NN and discussed the learning effect when performing a selecting operation during re-learning. In addition, we did not fix the weights of the policies reused at the time of transfer learning, and observed the learning effect when re-learning while selecting. In the case of not fixing the weights during transfer learning, convergence tends to be difficult. Therefore, whether it is necessary to fix weights when learning while selecting multiple policies was verified. In addition, the necessary conditions for adaptation to heterogeneous tasks and heterogeneous agents will be included in future prospects based on the learning results.

The source–task and the target–task were learned by the same task. In this experiment, the total reward obtained from the learning curve of each Episode versus the total reward, as well as the test results after learning, were evaluated. By this evaluation, we confirmed two points of effective learning and accurate learning.

B. Experimental Condition

For learning of the source–task and re-learning of the proposed method, a Deep Q–Network (DQN) built with library ChainerRL [19] was used. Experimental environment is build using Python, and for the learning task, CartPole and MountainCar, provided by OpenAI Gym [20], were adopted. Approximate models of the multiple methods used in the proposed method were designed randomly, assuming that a person cannot manually design an optimal hidden layer. The number of policies used in the proposed method was set to 10. In this experiment, the prototype used as a categorization perspective was set as the number of layers in the hidden layer of the policy.

For the DQN to be compared, the hidden layer was set to a 1 to 3 range, and the number of units per layer was set to 100. The number of units was approximately the same as the number of units in the hidden layer of the policy used for the proposed method. Statistic losses calculated for each step, which can be referenced by ChainerRL, were used for positive or negative judgment of policy transition. The learning

Algorithm 3 Transfer reinforcement learning with proposed method

Require: $\mathbb{A}$: Array that manages the activation value, M: SAP-Net, Π: Array that manages policies

$N =$is set as number of policies collected via source–task
$Episode = 1$
for $Episode = 1$ to Max Episode $+1$ **do**
  $reward = 0$
  $done = False$ ▷ *done* Fragment of granting rewards (True or False)
  $R = 0$
  $step = 1$
  $step_{store} = 1$ ▷ Fragment of store data for Experience Replay
  while *done* $\neq$ True and $step <$ Max step **do**
    if $step_{store} < N*$ Replay start size **then**
      Act a and store data$(s, a, s^{'}, r)$ to be used for Experience Replay when selecting each policy
    else
      Select policy π_i based on probability $P(\pi_i) = \dfrac{\exp(\mathbb{A}_i)}{\sum_{b=0}^{N-1} \exp(\mathbb{A}_b)}$
      Act a and training with π_i
      Calculates statistic loss $\mathbb{E}_t^{\pi_i}$
      $T_a = \mathbb{E}_{t+1}^{\pi_i} - \mathbb{E}_t^{\pi_i}$ ▷ Judgment positive or negative of policy transition
      if $T_a > 0$ **then** SPREAING$(i, \mathbb{A}, M, 0)$ ▷ Negative transfer
      else SPREAING$(i, \mathbb{A}, M, \mathcal{A})$ ▷ Positive transfer
      end if
      $R \leftarrow R + r$
      $step \leftarrow step + 1$
      $step_{store} \leftarrow step_{store} + 1$
    end if
  end while
  $step = 1$
  $step_{store} = 1$
end for

TABLE I. TASK AND PARAMETER SETTING

Learning task	CartPole	MountainCar
Max episode	5000	10000
Number of test	5	5
Activate function	ReLU	ReLU
Discount rate	0.99	0.99
Explorer	ϵ-greedy ($\epsilon = 0.03$)	ϵ-greedy ($\epsilon = 0.03$)
Optimizer	Adam ($\epsilon = 1e-3$)	Adam ($\epsilon = 1e-3$)
Replay buffer size	$1e+6$	$1e+6$
Replay start size	50	100

TABLE II. PARAMETER SETTING FOR PROPOSED METHOD

Parameter	Symbol	Value
Adjustment weight of $\mathbb{P}$	δ	-0.5
Initial weight of $\mathbb{P}$	ω_c	0.1
Activation function parameter	$\mathcal{A}$	1.0
Threshold	T_ω	1.0

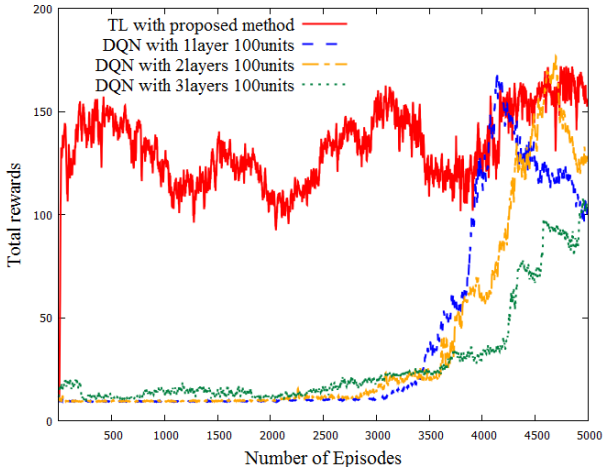

Fig. 5. Comparison with Learning Curve (CartPole).

conditions of DQN are shown in Table I. For parameters other than Table I, ChainerRL's initial setting was used. Episode per total rewards was averaged 5 times, test results were summarized 5 times and expressed as a learning curve and bar graph. The learning curves are indicated by the moving average value every 10 episodes in order to easily observe the increase or decrease of the total reward for each episode. Table II shows the parameters of the proposed method. The transfer learning algorithm of reinforcement learning used in the proposed method is shown in Algorithm 3.

C. Experimental Results

The learning curve of the CartPole task is shown in Fig. 5, and the test result after learning is shown in Fig. 6. The learning curve of the MountainCar task is shown in Fig. 7, and the test result after learning is shown in Fig.8. In Fig. 5 and Fig. 7, the red learning curve shows transfer learning (TL: Transfer Learning) using the proposed method. The blue learning curve represents the result of reinforcement learning (DQN: Deep Q–Network) of 100 units in one layer, the orange

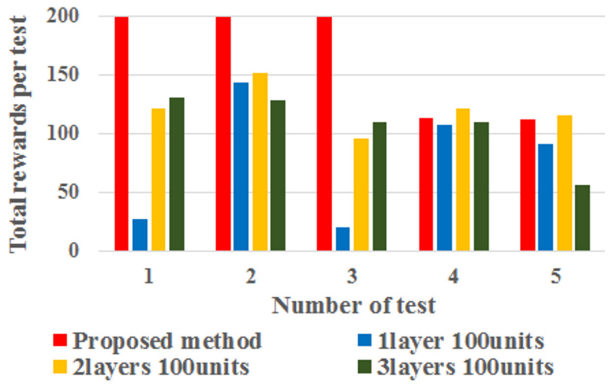

Fig. 6. Test Task After Learning (CartPole).

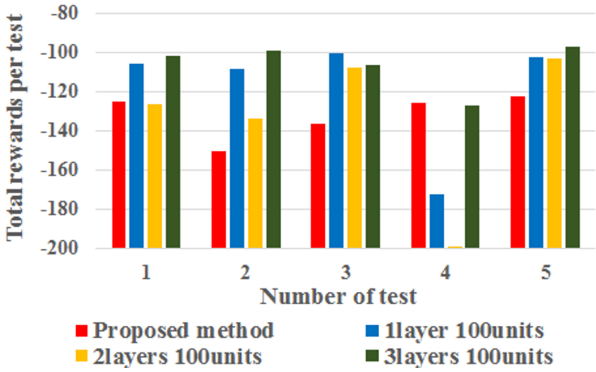

Fig. 8. Test Task After Learning (MountainCar).

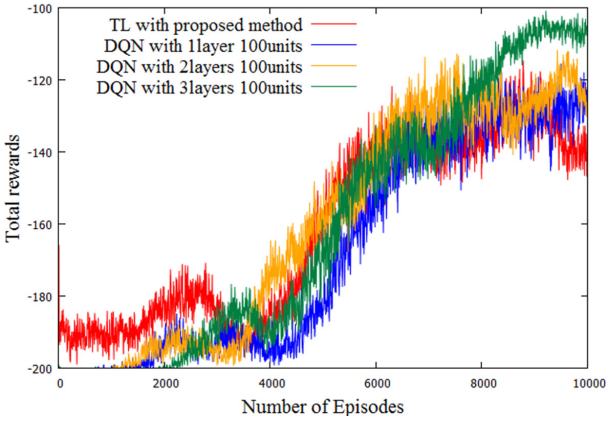

Fig. 7. Comparison with Learning Curve (MountainCar).

learning curve of 100 units in 2 layers, and the green learning curve of 100 units in three layers. The vertical axis represents the total rewards, and the horizontal axis represents the task execution number (Episode).

In Fig. 6 and Fig.8, the vertical axis represents the total rewards, and the horizontal axis represents the number of tests. The color of the bar graph shown in the test result corresponds to the color of the learning curve.

From the results shown in Fig. 5, it can be confirmed that high rewards can be earned from an Episode at the initial learning stage. From this result, it is understood that the learning is promoted by the proposed method. From the learning results of the three types of DQN to be compared, a phenomenon in which the reward acquisition amount declined was seen. This may be caused by the fact that the structure of the learning model is not optimized for the task. When learning from the beginning, in order to perform optimum learning, it is necessary to adjust the NN structure and its parameters to a specific task. From the test results of CartPole in Fig. 6, the proposed method confirmed that it is possible to acquire high rewards at high frequency. This is presumed to be due to re-learning of the policy with the highest priority being possible while selecting the policy. We confirmed that the reward decreased in the 4th and 5th tests. It is presumed that this was caused by the fluctuation of the teacher data value during the learning process. In this regard, it may be possible to cope with by fixing the weight of the approximate model for each policy. From the above results, it was possible to confirm the usefulness of the proposed method in CartPole.

From the results shown in Fig. 7, it can be confirmed that, although the amount of reward earned at the early stage of learning was large, the proposed method decreased the amount of compensation earned in the Episode near the end of learning, compared to the DQN. This result seems to be caused by being unable to perform the MountainCar task. In this task, judging positive or negative transition is carried out at the end of the task, while in the proposed method, the transfer judgment of the policy is carried out for each action. In order to increase the reward acquisition amount against this result, this may be solved by examining whether to perform the policy transition judgment for each Episode or to perform it every certain step number. In addition, by considering indices other than the loss for activation judgment, some conditions may be judged more effectively.

Considering the usable range of the proposed method, we think that it can be applied to tasks where timing of reward assignment is likely to be associated with task achievement condition. In future prospects, if SAP-Net includes policies for heterogeneous tasks and agents, we estimate that a new method is needed that takes into account the above applicable range. For that method, it is desirable to calculate selection candidates of policies by using categories, or to exclude policies presumed to be unnecessary for learning of the target–task. In the case of policies in heterogeneous agents, processing the data to enable reusability by the target–task through Inter task mapping is necessary.

IV. Conclusion

In this paper, we proposed a method to re-learn while choosing a policy as transfer learning. The method was implemented by using a spreading activation model and was verified by a computer experiment. From the results of the verification, usefulness was suggested in the transfer learning of the same task without manual evaluation by NN model design. In addition, as a countermeasure to the problem, we will examine judgment of appropriate positive and negative transition, and consider selection candidate calculation of policies using categories in transfer learning of heterogeneous tasks and agents. We also think that it is possible to develop

a method of applying Inter task mapping for related tasks by applying categories to this problem.

ACKNOWLEDGMENT

Part of this research was undertaken with the aid of JSPS Grant-in-Aid for Scientific Research (JP16K12493 and JP19K12173). We express our gratitude here.

REFERENCES

[1] T. Tongloy, S. Chuwongin, K. Jaksukam, C. Chousangsuntorn, and S. Boonsang, "Asynchronous deep reinforcement learning for the mobile robot navigation with supervised auxiliary tasks," in *2017 2nd International Conference on Robotics and Automation Engineering (ICRAE)*, 2017, pp. 68–72.

[2] T. Shimizu, R. Saegusa, S. Ikemoto, H. Ishiguro, and G. Metta, "Robust sensorimotor representation to physical interaction changes in humanoid motion learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 5, pp. 1035–1047, 2015.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

[4] M. E. Taylor, *Transfer in Reinforcement Learning Domains*, ser. Studies in Computational Intelligence. Springer, 2009, vol. 216.

[5] F. Fernández and M. Veloso, "Learning domain structure through probabilistic policy reuse in reinforcement learning," *Progress in Artificial Intelligence*, vol. 2, no. 1, pp. 13–27, 2013.

[6] T. Takano, H. Takase, H. Kawanaka, and S. Tsuruoka, "Transfer method for reinforcement learning in same transition model – quick approach and preferential exploration," in *2011 10th International Conference on Machine Learning and Applications and Workshops*, vol. 1, 2011, pp. 466–469.

[7] H. Kono, R. Katayama, Y. Takakuwa, W. Wen, and T. Suzuki, "Activation and spreading sequence for spreading activation policy selection method in transfer reinforcement learning," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 12, pp. 7–16, 2019.

[8] A. M. Collins, F. Elizabeth, and Loftus, "A spreading-activation theory of semantic processing," *Psychological Review*, vol. 82, no. 6, pp. 407–428, 1975.

[9] M. Han and B. Zhang, "Control of robotic manipulators using a cmac-based reinforcement learning system," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)*, vol. 3, 1994, pp. 2117–2122.

[10] Y.-P. Hsu, W.-C. Jiang, and H.-Y. Lin, "A cmac-q-learning based dyna agent," in *2008 SICE Annual Conference*, 2008, pp. 2946–2950.

[11] F. Li, F. Luo, Y. Gao, D. Qi, and J. Hu, "Research on fuzzy reinforcement learning algorithm for agents in grids," in *2009 Third International Symposium on Intelligent Information Technology Application Workshops*, 2009, pp. 336–339.

[12] X.-N. Wang, X. Xu, and H.-G. He, "Policy gradient fuzzy reinforcement learning," in *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826)*, vol. 2, 2004, pp. 992–995.

[13] H. S. Cho, "A study on the control of nonlinear system using growing rbfn and reinforcement learning," in *Third International Conference on Natural Computation (ICNC 2007)*, vol. 5, 2007, pp. 521–525.

[14] S. Li, L. Ding, H. Gao, Y. Liu, N. Li, and Z. Deng, "Reinforcement learning neural network-based adaptive control for state and input time-delayed wheeled mobile robots," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 11, pp. 4171–4182, 2018.

[15] Y. Koizumi, K. Niwa, Y. Hioka, K. Kobayashi, and Y. Haneda, "Dnn-based source enhancement self-optimized by reinforcement learning using sound quality measurements," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 81–85.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[17] T. Okuyama, T. Gonsalves, and J. Upadhay, "Autonomous driving system based on deep q learnig," in *2018 International Conference on Intelligent Autonomous Systems (ICoIAS)*, 2018, pp. 201–205.

[18] H. Sasaki, T. Horiuchi, and S. Kato, "A study on vision-based mobile robot learning by deep q-network," in *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, 2017, pp. 799–804.

[19] Chainerrl. [Online]. Available: https://github.com/chainer/chainer

[20] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016, cite arxiv:1606.01540. [Online]. Available: http://arxiv.org/abs/1606.01540