Paper:

Exploitation-Oriented Learning with Deep Learning – Introducing Profit Sharing to a Deep Q-Network –

Kazuteru Miyazaki

National Institution for Academic Degrees and Quality Enhancement of Higher Education
1-29-1 Gakuennishimachi, Kodaira, Tokyo 185-8587, Japan
E-mail: teru@niad.ac.jp

[Received March 21, 2017; accepted July 21, 2017]

Currently, deep learning is attracting significant interest. Combining deep Q-networks (DQNs) and Q-learning has produced excellent results for several Atari 2600 games. In this paper, we propose an exploitation-oriented learning (XoL) method that incorporates deep learning to reduce the number of trial-and-error searches. We focus on a profit sharing (PS) method that is an XoL method, and combine it with a DQN to propose a DQNwithPS method. This method is compared with a DQN in Atari 2600 games. We demonstrate that the proposed DQNwithPS method can learn stably with fewer trial-and-error searches than required by only a DQN.

Keywords: reinforcement learning, deep learning, deep reinforcement learning, profit sharing, deep Q-network

1. Introduction

Recently, deep learning has attracted considerable attention. The well-known deep Q-network (DQN) [1] successfully combines deep learning and Q-learning (QL) [2], the most representative reinforcement learning (RL) method. DQNs with QL have produced excellent results for several Atari 2600 games using the arcade learning environment [3].

QL is an RL method based on dynamic programming, and it requires numerous trial-and-error searches. In this study, we combine a DQN and profit sharing (PS) method [4, 5] to reduce the number of necessary trial-and-error searches. PS is one of the exploitation-oriented learning (XoL) [6] methods. XoL is an approach that treats reward and penalty signals independently, and enhances the successful experiences strongly to reduce the number of trial-and-error searches.

XoL has the following four features: (1) It learns more rapidly by strongly tracing the successful experiences. (2) It treats rewards and penalties as independent signals, allowing these signals to be handled more intuitively and easily than concrete values. (3) It does not achieve optimality efficiently, which can be acquired by multi-start [7] resetting all memory or reward scheduling. (4) It is strong in the class that exceeds the Markov decision processes (MDPs) because it is a Bellman-free method.

Therefore, we expect to reduce the number of trial-and-error searches by combining PS with a DQN. We refer to the proposed method as DQNwithPS. This method is compared with a DQN in Atari 2600 games. We demonstrate that the proposed DQNwithPS method can learn stably with fewer trial-and-error searches than a DQN.

2. Domain

2.1. Definition of Terms

In RL, a subject is commonly known as an *agent*. After receiving sensory inputs from the environment, an agent selects an *action* and performs it. *Rewards* and *penalties* are assigned based on a series of actions. A reward (or penalty) is awarded to a state or an action that satisfies (or does not satisfy) the purpose. The sensory inputs and the corresponding selected action comprise a unit, known as a *step*. Sensory inputs from the environment are referred to as *states*. A pair of state and action is known as a *rule*. If an agent selects action a_t in state s_t, the rule is described as $rule(s_t, a_t)$. Note that in some cases, the agent may have to select different actions for the same sensory input because the sensory input is incomplete; this is known as the *perceptual aliasing problem* [8], and several methods have been proposed to resolve it [9].

A sequence of steps from an initial state to a reward or penalty is known as an *episode*. If different rules are selected for the same state, i.e., if the episode contains a loop, the loop is referred to as a *detour*. A rule that is always present in a detour is called an *ineffective rule*; otherwise, it is known as an *effective rule*. If both effective and ineffective rules are related to the same sensory input, the ineffective rule should not be selected. If a rule that has previously been considered as an effective rule is always present a detour, it is called *type 2 confusion* [7]. A function that maps a state to an action is known as a *policy*. A policy is *deterministic* if there is only one action for each state, and a policy is *rational* if all the expected rewards per action are positive. The policy that maximizes the expected reward per action is the *optimal policy*.

2.2. Q-Learning

Each time an agent selects an action, QL learns by updating the *Q-value* that is a function representing the value of the action. When an agent selects action a_t in state s_t, receives reward r_t, and transits to next state s_{t+1}, $Q(s_t,a_t)$, the Q-value of $rule(s_t,a_t)$ is updated by the following equation:

$$Q(s_t,a_t) = (1-\alpha)Q(s_t,a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1},a')) \quad . (1)$$

where α is the learning rate and γ is the discount rate.

QL ensures the determination of the optimal policy in the MDPs. We can obtain the optimal policy if the corresponding agent selects an action that has the maximum Q-value in the state, after the Q-value has converged, by reducing learning rate α according to a previously proposed theorem [2]. However, this theorem does not ensure any rationality before the Q-value convergence. Furthermore, QL typically requires numerous trial-and-error searches to attain convergence because it must interact with the environment not only to identify the unknown environment, but also to update the Q-value [10].

2.3. Profit Sharing

PS learns a rational policy by propagating a reward in an episode backward when the reward is awarded. Assume that action a_t was used in state s_t at time t, and let $Q(s_t,a_t)$ be a reward shared in $rule(s_t,a_t)$. In this study, we use the following geometric decreasing function to propagate the reward backward:

$$Q(s_t,a_t) = \lambda^{N-t}R, \quad t = 1,2,\ldots,N,\ 0 < \lambda < 1, \quad (2)$$

where R is the reward value, N is the episode length, and λ is the *discount rate*. A function that propagates a reward is known as a *reinforcement function*.

Equation (2) is derived from *the rationality theorem of PS* [4, 5], that is the necessary and sufficient condition required to learn a rational policy in a class in which only one type of reward is present and type 2 confusion is absent. If we set $\lambda = \frac{1}{Types\ of\ actions}$ in Eq. (2), we can satisfy the theorem. In contrast, the *rationality theorem of PS in multi-agent learning* [11] is the necessary and sufficient condition for distributing a reward to an agent that does not receive a direct reward.

PS is an XoL method that strongly enhances the successful experiences. Therefore, generally, PS can learn a rational policy with fewer actions than QL. Furthermore, it can ensure that the rational policy can be acquired in a class without any type 2 confusion. Although the RL methods based on dynamic programming such as QL can guarantee acquisition of the optimum policy in the MDPs that are a part of a class without type 2 confusion, the rationality of a class that exceeds the MDPs is unknown.

3. Deep Q-Network

In a DQN, the Q-value is approximated using a convolutional neural network (CNN). A DQN has been applied to seven Atari 2600 games [1], outperforming the existing methods, and achieving a score higher than that achieved by human experts for some games.

In a DQN, the game screen is input directly to the CNN. Therefore, it is typically associated with the perceptual aliasing problem. A DQN inputs the last four frames[1] into a CNN simultaneously to reduce the effects of this problem. The perceptual aliasing problem in DQNs has been investigated. For example, in a previous study [12], ten frames were input into the CNN.

In this study, the network configuration is the same as a previously reported configuration [1]. To learn the network parameters, the DQN uses *replay memory* that stores *the experience* comprising of four tuples (s_t,a_t,r_t,s_{t+1}) when the agent selects action a_t in state s_t, receives a reward r_t, and transits to next state s_{t+1}. The four tuples in the replay memory continues to store all the experiences that the agent had to order. If the required storage area exceeds the available memory, the oldest experience is deleted. Minibatch (MB) learning has been performed to update the CNN parameters. First, a certain number of experiences is sampled randomly from the replay memory. Subsequently, for each experience, the gradient is calculated when $r_j + \gamma \max_{a'} Q(s_{j+1},a')$ is set to a target value. Finally, the CNN parameters are updated according to the calculated gradients.

4. Proposed DQNwithPS

4.1. Proposed Method and Limitations of DQN

Typically, QL requires numerous trial-and-error searches. Here the DQN starts learning the CNN parameters after 100 frames; however, it is generally difficult for the Q-values to convergence in 100 frames. Consequently, the CNN parameters are updated without the Q-value convergence in the early learning states. This is inefficient learning from the perspective of RL.

In this study, we aim to reduce the number of actions required to learn the CNN parameters. We combine PS that can rapidly provide a rational policy, and a DQN.

In general, a game problem is considered to be simulation-based learning. Numerous trial-and-error searches are not problematic when an effective simulator is used. However, an effective simulator is not always available; therefore it is important to take good care of the valuable experience. This is the main objective of our study including all the XoL methods.

4.2. DQNwithPS

To reduce the number of trial-and-error searches, we propose to employ PS with a DQN each time an agent obtains a reward as shown in **Fig. 1**.

1. A single frame corresponds to selecting a single action.

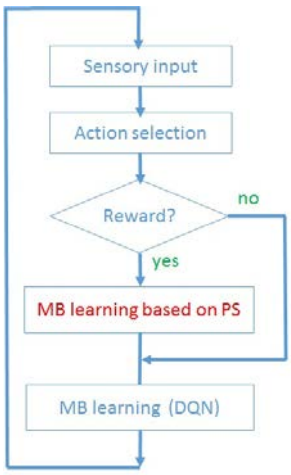

Fig. 1. DQNwithPS algorithm.

DQNwithPS uses the following "MB learning based on PS." First, the value obtained by Eq. (2) is distributed to all the rules in the episodes in which the agent obtained a reward, and the value is stored in the replay memory. Second, a certain number of experiences is sampled randomly from the replay memory corresponding to the episode. Third, for each experience, the gradient is calculated when $r_j + \gamma \max_{a'} Q(s_{j+1}, a')$ is set to the target value. Finally, the CNN parameters are updated according to these gradients. This method follows MB learning in the DQN. Though the amount of computational time with the DQNwithPS method temporarily increases when a reward is acquired, it is much less than that for requiring a target in each action selection.

The main part of the DQNwithPS method is the same as that of a DQN, and the only difference from the DQN is to run the above "MB learning based on PS" when an agent receives a reward. In contrast, both the DQN and DQNwithPS handle a penalty through QL.

There is a tradeoff between the learning speed and learning result. The same tradeoff will occur in PS MB learning. Although a correct answer can be obtained by repeating MB learning, the obtained result will not necessarily correspond to a desired solution. An appropriate γ is expected to vary depending on the problem and therefore, we confirm the behavior through numerical experiments.

4.3. Related Works

Methods to improve the DQNs often focus on improving the score [13]. A previous study attempted to reduce the number of trial-and-error searches by improving the search method of the environment [14]. Other studies have attempted to increase the speed of learning by using a model [15] or considering the QL discount rate γ [16]. In addition, another previous study focused on the perceptual aliasing problem [12]. Although these earlier works were based on QL, we focus on PS, a different learning approach for refining the learning speed by combining PS and a DQN. This is the primary contribution of this paper. The proposed DQNwithPS method is easy to combine with previous methods that attempt to enhance DQNs because it simply involves adding PS to the DQN and does not require changing the essence of the DQN.

5. Numerical Experiments

5.1. Experimental Setting

The sizes of the MB and replay memory are 32 and 1,000,000 frames, respectively. Note that these values were taken from the literature [1]. Discount rate for QL, γ, is 0.99. We use the geometric decreasing function in Eq. (2) as the reinforcement function of PS with λ as the discount rate.

Although DQNs have been evaluated for seven games [1], in this study, the DQNwithPS method is evaluated for two games called Pong and Breakout. In Pong, when an agent hits a ball, but its opponent agent cannot, the agent receives a point in the game and is awarded a reward of value 1.0. However, when the agent cannot hit a ball, it loses a point and suffers a penalty of value -1.0. Once the total points of either agent reaches 21, the game ends, and score of the game is calculated based on the difference between the scores of the agent and its opponent. Therefore, the score of Pong is an integer value between -21 and 21.

In Breakout, when the agent hits a ball and turns off a block, the agent receives a point in the game and is rewarded a value of 1.0. However, there is no penalty. The agent has five balls at the beginning of the game. When the agent hits a ball, but cannot turn off a block, the ball is lost. After the number of balls reaches zero, the game ends and score of the game is calculated based on the number of blocks turned off. Therefore, the score is a natural number. Thus, the values of a reward and penalty are decided by a score in a DQN. Consequently, there is no penalty in Breakout because the score of Breakout cannot be negative.

We confirm the performance of the DQNwithPS method by varying λ. The learning result is evaluated in terms of the score. We obtain ten scores without learning for 100 episodes of Breakout and ten episodes of Pong, and we compare the methods by their average values.

The action is selected using an ε-greedy strategy ($\varepsilon = 0.01$) in the evaluation, in which a random action and the action with the maximum Q-value are selected with a probability of 0.01 and 0.99, respectively. Although the proposed DQNwithPS method does not change ε in the experiment, the DQN changes ε in the ε-greedy strategy according to the following strategy that was reported in the literature [1].

- $\varepsilon = 0.1$ (if the number of actions exceeds 1,000,000)
- $\varepsilon = 1.0 - 0.9 * \dfrac{The\ number\ of\ actions}{1000000}$ (otherwise)

We set $\varepsilon = 0.01$ in the DQNwithPS method to take ad-

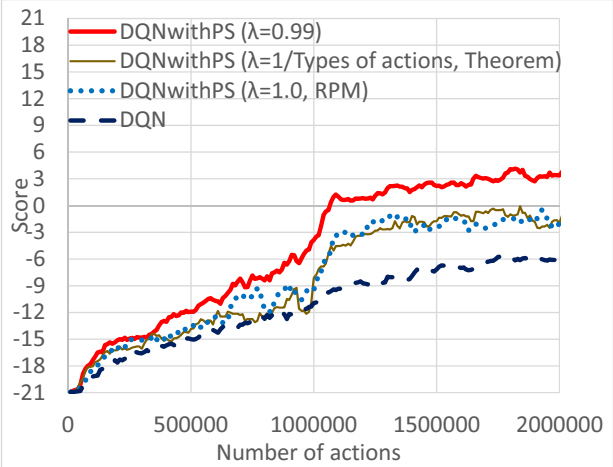

Fig. 2. Pong results when varying γ.

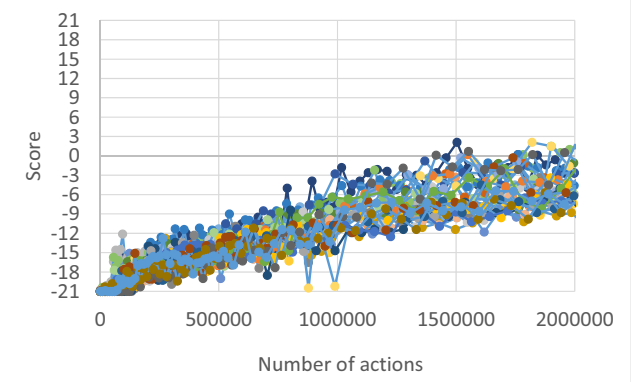

Fig. 3. Ten Pong results for DQN.

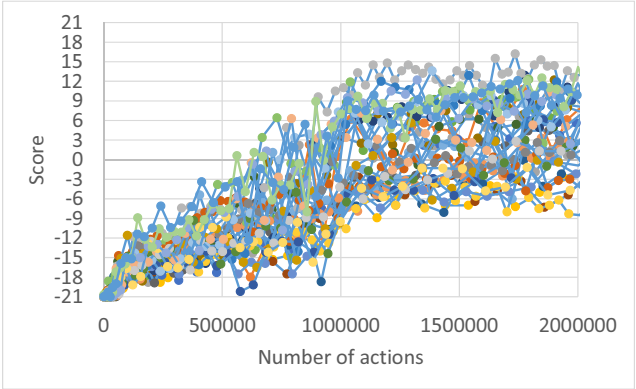

Fig. 4. Ten Pong results for DQNwithPS ($\lambda = 0.99$).

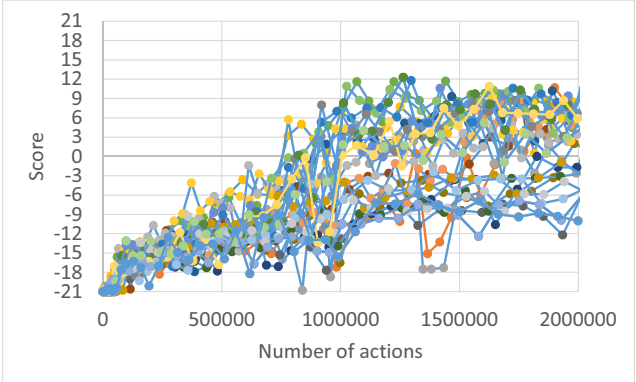

Fig. 5. Ten Pong results for DQN when using fixed $\varepsilon = 0.01$.

vantage of the experience-oriented aspects of PS. Results when performing the same setting in DQN are discussed later.

We use the 64-bit version of Ubuntu 14.04 as the operating system and the Caffe deep learning framework. We refer to two web sites to construct the experimental environment[2,3].

5.2. Experimental Results

Each experiment was performed 30 times by varying λ. The results with Pong are shown in **Figs. 2–6**. **Fig. 7** shows the result with Breakout. These figures depict the score of each game plotted as a function of the number of actions that is equal to the number of frames.

Figure 2 shows the results for varying λ. λ is considered as 1.00, 0.99, and $\frac{1}{Types\ of\ actions}$ to satisfy the rationality theorem of PS. Note that $\lambda = 1.0$ corresponds to the rational policy making algorithm (RPM) [7]. **Figs. 2** and **6** include the results for the DQN. Here, the score is the average of the 30 experiments.

Figures 3–5 show simultaneously the results of the 30 experiments. **Fig. 3** depicts the results for the DQN, and **Fig. 4** displays the results for DQNwithPS, where $\lambda =$ 0.99. **Fig. 5** shows 30 results for the DQN when using fixed $\varepsilon = 0.01$. **Fig. 6** shows the comparison of the DQN at fixed $\varepsilon = 0.01$ with the DQNwithPS method. **Fig. 7** illustrates the results using Breakout, where $\lambda = 0.99$.

5.3. Discussion

5.3.1. Pong Results

Considering λ, as shown in **Fig. 2**, the DQNwithPS method learns with fewest actions when $\lambda = 0.99$. In contrast, the effectiveness of the DQNwithPS method is degraded with RPM ($\lambda = 1.0$) and the function that satisfies the rationality theorem of the PS. In general, the probability that the same sensory input will occur is low in a continuous image input. Therefore, in the learning method, e.g., RPM, it is possible that all the selected actions will be equally evaluated as effective. With $\lambda = 1.0$, the original purpose to provide a correct answer quickly is not achieved, and an evaluation of an action that has been prioritized by QL may be influenced negatively by the RPM. The result for the RPM shown in **Fig. 2** is a typical example of this.

By using the function that satisfies the rationality theorem of PS, it is unlikely that QL will be inhibited. However, when the reward value is discounted by $\lambda = \frac{1}{Types\ of\ actions}$, the value becomes exponentially small.

2. http://eiji-kb.hatenablog.com/entry/2015/07/29/200733
3. http://d.hatena.ne.jp/muupan/20141021/1413850461

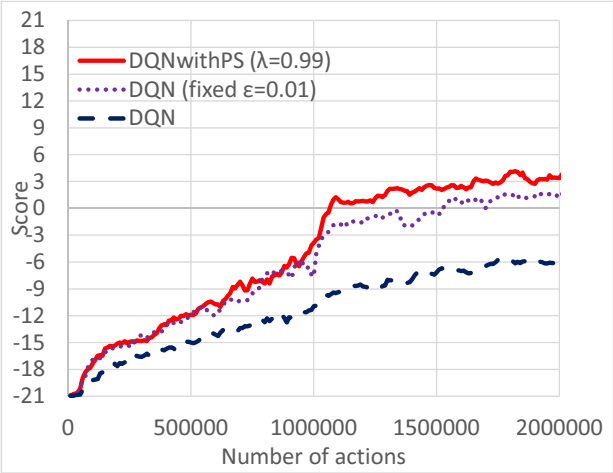

Fig. 6. Comparison with DQN, when DQN uses fixed $\varepsilon = 0.01$ and DQNwithPS ($\lambda = 0.99$).

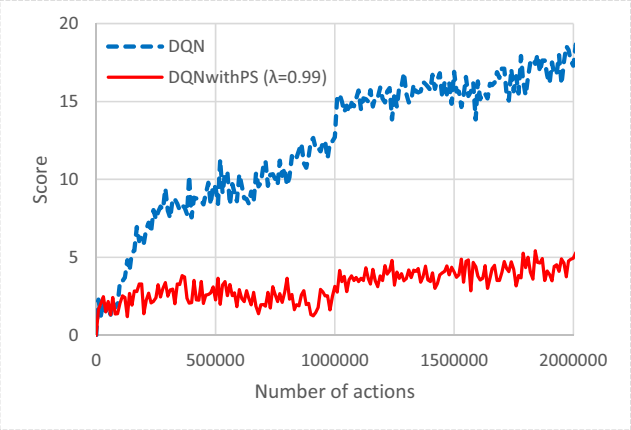

Fig. 7. Breakout results.

Table 1. Results of comparison with human expert.

	DQN	DQNwithPS		
		1.0	0.99	$\frac{1}{Types\ of\ actions}$
1	530367	490494	404528	358846
2	845544	843506	736109	646588
3	499987	1014250	221412	888664
4	998535	476148	246643	490751
5	648270	171043	523434	882003
6	1225023	1244029	757289	547759
7	1214996	826361	820770	696606
8	808195	724580	372705	333761
9	544656	311513	143462	607137
10	680979	204513	746697	570331
11	974645	595113	130771	1018803
12	1154368	320176	559304	306369
13	1072281	881852	781087	497718
14	792982	488177	515961	870115
15	645373	881567	453819	144197
16	1203775	680876	170883	703009
17	1029416	419121	304579	1140153
18	1131848	807714	132448	1207221
19	710952	546669	175562	375225
20	500578	398636	386470	296743
21	1279726	119043	348945	492078
22	1023521	395713	466793	390370
23	829581	466844	312022	298747
24	1031356	634978	139027	565379
25	1031776	266567	321616	180677
26	535967	123553	267515	482742
27	963071	484691	432116	151358
28	931305	352748	423904	314463
29	802330	98642	648798	360756
30	855017	286051	980221	1280032
ave.	883000	519000	431000	570000

Therefore, even with the usefulness of a theorem to ensure the rationality, in a practical sense, the learning speed is not enhanced significantly. The rationality theorem of PS states that it is possible to learn a rational policy in an environment where the same sensory input will be obtained. However, as mentioned previously, in this problem, the possibility that the same sensory input is obtained is low. Therefore, in this problem, the discount of a reward such that $\frac{1}{Types\ of\ actions}$, is not required.

The rationality theorem of PS is a condition for an arbitrary environment. This condition will be a highly strict condition for Pong. We, therefore, believe that the case of $\lambda = 0.99$ is a looser condition than the theorem as it yields the best result with Pong. Consequenlty, we consider that $\lambda = 0.99$ is the most effective value for Pong. In the future, we plan to verify this with more games to confirm the generality.

30 experimental results

It can be seen from the 30 experimental results shown in **Figs. 3** and **4** that although the variation in the DQN results is small, the results obtained with DQNwithPS relatively vary more. PS learns a policy that is strongly dependent on the first obtained reward. These results are owing to the exploitation-oriented aspects of PS. Although there is a variation between the experiments in terms of the number of actions required, the score improves, and any of the results are better thatn those of the DQN. After completion of the feature extraction of the game screen, the effectiveness of PS will be apparent.

Comparison with human expert

According to a previous report [1], the score for human experts with Pong was -3. We compared it with the number of actions required to first arrive at a score based on the ten best performances of scores from our 30 experimental results. The corresponding results of the DQN and DQNwithPS ($\lambda = 1.0, 0.99$, and $\frac{1}{Types\ of\ actions}$) are presented in **Table 1**. For Pong, DQN requires approximately $883,000$ average actions to arrive at a score of -3. In contrast, the proposed DQNwithPS requires approximately $519,000$, $431,000$, and $570,000$ average actions for $\lambda = 1.00, 0.99$, and $\frac{1}{Types\ of\ actions}$, respectively. Thus, DQNwithPS outperforms the DQN by factors of approximately 1.65, 2.05, and 1.50, respectively.

Table 2. Results of t-test.

actions ($\times 10^6$)	0.9	1.0	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2.0
t-value	1.764	5.193	4.226	3.650	3.966	5.319	3.963	2.818	4.431	3.649	2.067	3.391

About fixed ε

From **Fig. 6**, we can clearly see the difference between the DQNwithPS and DQN ($\varepsilon = 0.01$) results after 900,000 actions. We conducted a t-test every 100 thousand actions between DQNwithPS ($\lambda = 0.99$) and DQN (fixed $\varepsilon = 0.01$) shown in **Figs. 4** and **5** after 900,000 actions. We have confirmed the normality and homoscedasticity as a pre-test before applying the t-test. We show the t-values in **Table 2**. The calculation of the t-test demonstrated that DQNwithPS ($\lambda = 0.99$) is statistically more significant and superior to DQN (fixed $\varepsilon = 0.01$) at $p = 0.05$ level (1.697). This implies that DQNwithPS still outperforms DQN ($\varepsilon = 0.01$) in terms of the score.

The result also concerns with a tradeoff problem between exploitation and exploration. The proposed DQNwithPS method is one of the XoL methods with emphasis on the exploitation aspect. PS can guarantee a rationality when the agent obtains a reward once. This implies that the fixed ε is suitable for PS. Therefore, DQNwithPS can use the fixed ε to reduce the trial-and-error searches. Conversely, DQN must change ε to obtain a stable behavior because QL does not ensure any rationality before converging the Q-value. Therefore, the DQN must control ε to obtain a stable behavior. This confirms that DQNwithPS performs better with fewer trial-and-error searches than DQN ($\varepsilon = 0.01$).

5.3.2. Breakout Results

For Breakout, based on the knowledge obtained with Pong, we set $\lambda = 0.99$. It can be seen in **Fig. 7**, the final score of DQNwithPS is less than that of DQN.

The reward in Pong is awarded to an agent as a point, and the penalty is assigned to an agent as a loss of a point. This directly increases the final score. This implies that the agent continues to obtain a reward as the agent never loses the game when it keeps hitting a ball. In Breakout, the reward is given to an agent by removing a block through hitting a ball, and there is no penalty. Therefore, PS cannot always provide an approach to remove a better block because it aims to receive rewards in fewer trial-and-error searches rather than to acquire the optimal policy. Conversely, QL may learn a policy that can remove a block better because QL always pursues an optimal policy.

PS should be used in a task where a reward design is connected to our purpose directly. Specifically, we should take the reward setting in which our purpose is achieved by acquiring a reward. Pong is such an example. If we expect to obtain a better reward for PS in a reward setting that aims to maximize the expected reward per action, we should introduce a multi-start [7] or reward scheduling.

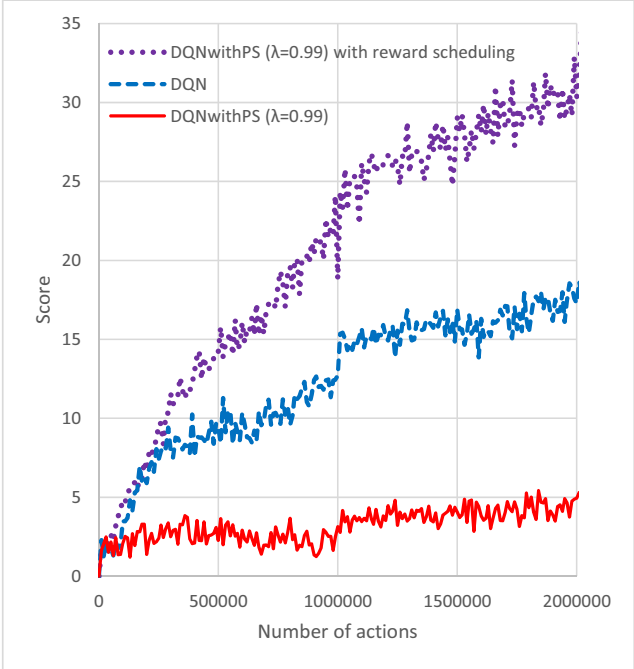

Fig. 8. Breakout results with reward scheduling.

As one of the reward scheduling, we can consider the following method:

- If the number of times that a block is erased without losing a ball increases, MB learning based on PS is added to the DQN.
- Otherwise, it is not added to the DQN.

This suggests that "reward" in **Fig. 1** changes to "reward and the number of times that a block is erased without losing a ball increase."

The result is shown in **Fig. 8**. The method with the reward scheduling shows a significant improvement. Thus, we have confirmed the effectiveness of DQNwithPS through numerical experiments.

6. Conclusion

In this paper, we have proposed an XoL that incorporates deep learning. The proposed DQNwithPS method was compared with a DQN in Atari2006 games. Through numerical experiments, we demonstrated that the proposed DQNwithPS method can learn with fewer trial-and-error searches than only a DQN.

The proposed method can be easily combined with various DQNs. We will be conducting such experiments in

the future. We will also perform several experiments using other games in the future. Currently, we are implementing a combination of the *Expected Failure Probability* (EFP) method [17] and PS. EFP is an XoL method that avoids a penalty by estimating an expected failure probability of a rule. In this case, it is necessary to prepare each CNN for PS and EFP. We believe that the proposed method will benefit from the EFP method. Furthermore, we can easily combine our proposed method to some variation of the DQN.

In addition to the improvements afforded by such a method, we plan to apply the proposed DQNwithPS method to other issues such as a keepaway task [18, 19], biped walking robot [20], course classification support system [21], and consciousness system [22].

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 26330267 and 17K00327.

References:

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," NIPS Deep Learning Workshop 2013, 2013.

[2] C. J. H. Watkins and P. Dayan, "Technical note: Q-learning," Machine Learning, Vol.8, pp. 55-68, 1992.

[3] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," J. of Artificial Intelligence Research, Vol.47, pp. 253-279, 2013.

[4] K. Miyazaki, M. Yamamura, and H. Kobayashi, "A Theory of Profit Sharing in Reinforcement Learning," Trans. of the Japanese Society for Artificial Intelligence, Vol.9, No.4, pp. 580-587, 1994 (in Japanese).

[5] K.Miyazaki, M. Yamamura, and S. Kobayashi, "On the Rationality of Prot Sharing in Reinforcement Learning," Proc of the 3rd Int. Conf. on Fuzzy Logic, Neural Nets and Soft Computing, pp. 285-288, 1994.

[6] K. Miyazaki and S. Kobayashi, "Exploitation-Oriented Learning PS-r$^{\#}$," J. Adv. Comput. Intell. Intell. Inform., Vol.13, No.6, pp. 624-630, 2009.

[7] K. Miyazaki and S. Kobayashi, "Learning Deterministic Policies in Partially Observable Markov Decision Processes," Proc. of the 5th Int. Conf. on Intelligent Autonomous System, pp. 250-257, 1998.

[8] L. Chrisman, "Reinforcement learning with perceptual aliasing: The perceptual distinctions approach," Proc. of the 10th National Conf. on Artificial Intelligence, pp. 183-188, 1992.

[9] M. T. Spaan, "Partially Observable Markov Decision Processes," Reinforcement Learning, Chapter 12, pp. 387-414, Springer-Verlag Berlin Heidelberg, 2012.

[10] K. Miyazaki, M. Yamamura, and S. Kobayashi, "k-Certainty Exploration Method : An Action Selector to identify the environment in reinforcement learning," Artificial Intelligence, Vol.91, No.1, pp. 155-171, 1997.

[11] K. Miyazaki and S. Kobayashi, "Rationality of Reward Sharing in Multi-agent Reinforcement Learning," New Generation Computing, Vol.19, No.2, pp. 157-172, 2001.

[12] M. Hausknecht and P. Stone, "Deep Recurrent Q-learning for Partially Observable MDPs," arXiv:1507, 2015.

[13] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A. D. Maria, M. Suleyman, C. Beattie, S. Petersen, S. Legg, V. Mnih, and D. Silver, "Massively Parallel Methods for Deep Reinforcement Learning," ICML Deep Learning Workshop, 2015.

[14] I. Osband, C. Blundell, A. Pritzel, and B. V. Roy, "Deep Exploration via Bootstrapped DQN," arXiv:1602, 2016.

[15] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous Deep Q-learning with Model-based Acceleration," arXiv:1603, 2016.

[16] V. Francois-Lavet, R. Fonteneau, and D. Emst, "How to Discount Deep Reinforcement Learning: Towards New Dynamic Strategies," NIPS 2015 Deep Reinforcement Learning Workshop, 2015.

[17] K. Miyazaki, H. Muraoka, and H. Kobayashi, "Proposal of a Propagation Algorithm of the Expected Failure Probability and the Effectiveness on Multi-agent Environments," SICE Annual Conf. 2013, pp. 1067-1072, 2013.

[18] P. Stone, R. S. Sutton, and G. Kuhlamann, "Reinforcement Learning toward RoboCup Soccer Keepaway," Adaptive Behavior, Vol.13, No.3, pp. 165-188, 2005.

[19] T. Watanabe, K. Miyazaki, and H. Kobayashi, "A New Improved Penalty Avoiding Rational PolicyMaking Algorithm for Keepaway with Continuous State Spaces," J. Adv. Comput. Intell. Intell. Inform., Vol.13, No.6, pp. 675-682, 2009.

[20] S. Kuroda, K. Miyazaki, and H. Kobayashi, "Introduction of Fixed Mode States into Online Reinforcement Learning with Penalties and Rewards and its Application to Biped Robot Waist Trajectory Generation," J. Adv. Comput. Intell. Intell. Inform., Vol.16, No.6, pp. 758-768, 2013.

[21] K. Miyazaki and M. Ida, "Proposal and Evaluation of the Active Course Classification Support System with Exploitation-oriented Learning," Lecture Notes in Computer Science, Vol.7188, pp. 333-344, 2012.

[22] K. Miyazaki and J. Takeno, "The Necessity of a Secondary System in Machine Consciousness," Procedia Computer Science, Vol.41, pp. 15-22, 2014.

Name:
Kazuteru Miyazaki

Affiliation:
Research Department, National Institution for Academic Degrees and Quality Enhancement of Higher Education

Address:
1-29-1 Gakuennishimachi, Kodaira-city, Tokyo 187-8587, Japan

Brief Biographical History:

1996- Assistant Professor, Tokyo Institute of Technology
1998- Research Associate, Tokyo Institute of Technology
1999- Associate Professor, National Institution for Academic Degrees
2000- Associate Professor, National Institution for Academic Degrees and University Evaluation
2016- Associate Professor, National Institution for Academic Degrees and Quality Enhancement of Higher Education

Main Works:

- K. Miyazaki, "Exploitation-oriented Learning PS-r#," J. Adv. Comput. Intell. Intell. Inform. (JACIII), Vol.13, No.6, pp. 624-630, 2009.
- K. Miyazaki, M. Yamamura, and H. Kobayashi, "A Theory of Profit Sharing in Reinforcement Learning," Trans. of the Japanese Society for Artificial Intelligence, Vol.9, No.4, pp. 580-587, 1994 (in Japanese).

Membership in Academic Societies:

- The Japanese Society for Artificial Intelligence (JSAI)
- The Society of Instrument and Control Engineers (SICE)
- Information Processing Society of Japan (IPSJ)
- The Japan Society of Mechanical Engineers (JSME)
- The Robot Society of Japan (RSJ)
- The Institute of Electrical Engineers of Japan (IEEJ)
- Japanese Association of Higher Education Research