Generating Intrinsic Rewards by Random Recurrent Network Distillation

ZeFeng Xu[1,a)] Koichi Moriyama[1] Tohgoroh Matsui[2] Atsuko Mutoh[1] Nobuhiro Inuzuka[1]

Abstract: Exploration in sparse reward environments pose significant challenges for many reinforcement learning algorithms. Rather than solely relying on extrinsic rewards provided by environments, many state-of-the-art methods generate intrinsic rewards to encourage the agent explore the environments. However, we found that existing models fall short in some environments, where the agent must visit a same state more than once. Thus, we improve an existing model to propose a novel type of intrinsic exploration bonus which will reward the agent when a new sequence is discovered. The intrinsic reward is the error of a recurrent neural network predicting features of the sequences given by a fixed randomly initialized recurrent neural network. Our approach performs well in some Atari games where conditions must be fulfilled to develop stories.

Keywords: Curiosity-driven Exploration, Recurrent Network, Reinforcement Learning, Deep Learning, Intrinsic Rewards

1. Introduction

With the development of artificial intelligence, people have increasingly pursued the performance of learning algorithms. As one of the important branches of machine learning, reinforcement learning does not need to be familiar with the mathematical model of the problem when solving problems, nor does it need to manually provide a large amount of labeled training data. Its trial-and-error learning pattern is also very close to the human learning pattern. It has strong versatility and is considered to be one of the important foundations of artificial intelligence. Coming out of deep learning has brought the performance of reinforcement learning to a higher level. The reinforcement learning method combined with deep learning is called deep reinforcement learning. Deep reinforcement learning combines the decision-making ability of reinforcement learning and the perception ability of deep learning [1][2].

Reinforcement learning algorithms learn policy through artificially designed rewards. When the reward is very sparse, reinforcement learning is likely to fail to learn. However, in many real-world problems, rewards to the agent are extremely sparse, or absent altogether. Therefore, exploration in sparse reward environments pose significant challenges for many reinforcement learning algorithms. Rather than solely relying on rewards provided by environments, many state-of-the-art methods generate intrinsic rewards to encourage the agent to explore the environments. For example, Random Network Distillation (RND) [3] emboldens the agent to visit new states, Intrinsic Curiosity Module (ICM) [4] encourages the agent to learn environment dynamics and Sequential Intrinsic Reward Generator (SRG) [5] encourages the agent to visit new sequences. This kind of intrinsic reward comes from the imitation of human learning skills using intrinsic motivation [6].

In these methods, rewards are generated by evaluating the novelty of states, but the novelty of the order of states is not considered. For SRG, the teacher signal provided by the target network is the features of the states rather than the features of the sequences. Therefore, SRG fails to learn the sequential information as intended. The purpose of our research is to improve the performance of intrinsic motivation methods by proposing a novelty exploration mechanism. In this work, we improve the SRG algorithm to propose a novel type of intrinsic exploration bonus. The intrinsic reward is the error of a recurrent neural network predicting features of the sequences given by a fixed randomly initialized recurrent neural network.

2. Deep Reinforcement Learning in Hard Exploration Environments

The premise of reinforcement learning is that the agent has no prior knowledge of the environment. That is, the agent does not know what state it will transition to and what reward it will get when taking a certain action in a certain state. In addition, it can be seen from Bellman's equation that unless rewarded, the reinforcement learning agent cannot learn anything. Therefore, the agent is very difficult to learn in an environment where rewards are very sparse. Especially in deep reinforcement learning, this problem becomes more prominent. Because the dimension of the state to be processed by deep reinforcement learning is very large, general exploration methods, such as ϵ-*greedy*, are very inefficient. This section will describe how to solve this problem.

[1] Department of Computer Science, Graduate School of Engineering, Nagoya Institute of Technology, Nagoya,Japan
[2] Department of Clinical Engineering, College of Life and Health Sciences, Chubu University, Kasugai, Aichi, Japan
[a)] z.xu.089@nitech.jp

2.1 Intrinsic Motivation

Psychologists distinguish between extrinsic motivation and intrinsic motivation. Extrinsic motivation refers to doing something as a result of certain rewards, while intrinsic motivation refers to doing something voluntarily due to a certain intrinsic enjoyment. Intrinsic motivation allows organisms to explore the environment without clear rewards. These activities favor the development of broad competence rather than being directed to more externally directed goals [7]. Intrinsic motivation may help guide the exploration of reinforcement learning agents, especially in environments where the extrinsic feedback is sparse or missing altogether [8]. One of the most effective intrinsic motivation methods in reinforcement learning agents is the curiosity-based method that encourages the agent to learn about the environment dynamics.

2.2 Curiosity-Driven Exploration

Curiosity-driven exploration encourages agents to explore the environment through bonuses to learn about the dynamics of the environment. Curiosity can be formulated as the error or uncertainty of predicting the consequences of an agent's actions [9]. Pathak formulated curiosity as the error in an agent's ability to predict the consequence of its own actions in a visual feature space learned by a self-supervised inverse dynamics model [4]. Burda formulated curiosity as the error of a neural network predicting features of the observations given by a fixed randomly initialized neural network [3]. In the next section, we will introduce the Burda's method.

2.3 Random Network Distillation (RND)

In order to implement the curiosity exploration, it is necessary to record the visited states. However, since deep reinforcement learning assumes that the number of dimensions of the state is large, it is difficult to record all of them. Therefore, the following prediction errors framework is generally used.

$$\hat{y}_{t+1} = f(s_t, a_t) \tag{1}$$

$$i_t = \|\hat{y}_{t+1} - y_{t+1}\|^2 \tag{2}$$

where $\hat{y}_{t+1}$ is the predicting feature of next state, y_{t+1} is the real feature of next state, i_t is the intrinsic reward and f is the Deep Neural Network (DNN). Considering the framework, the state of frequent input is easier to predict, so the error will be small, and the intrinsic reward also will be small. On the contrary, the state of infrequent input is difficult to predict, so the error becomes larger and the intrinsic reward also becomes larger. In general, prediction errors can be attributed to a number of factors [3]:

(1) If the predictor cannot be generalized from the examples previously seen, the prediction error is high. Then, the novel experience corresponds to a high prediction error.
(2) Since the prediction target is stochastic, the prediction error is high.
(3) Prediction error is high because information necessary for the prediction is missing, or the model class of predictors is too limited to fit the complexity of the target function.

Factor 1 is a useful source of error since it quantifies the novelty of experience, whereas Factors 2 and 3 cause the noisy-TV problem [10].

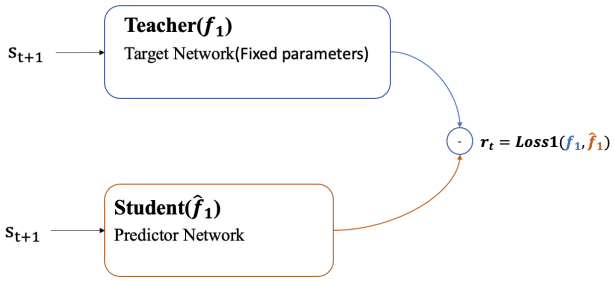

Fig. 1 Random Network Distillation (RND)

RND [3] can avoid Factors 2 and 3 with a new exploration bonus derived by predicting an output from a fixed and randomly initialized neural network. The intrinsic reward calculation process is shown in the Fig. 1. RND uses two DNNs to calculate the intrinsic reward as follows.

$$i_t = \frac{1}{N}\|\hat{f}(s_{t+1}) - f(s_{t+1})\|^2 \tag{3}$$

where N is dimension, $f(s_{t+1})$ is the *target network* and $\hat{f}(s_{t+1})$ is the *predictor network*. In other words, the output of the target network is used as the teacher signal, and the predictor network learns this signal. The advantage of using RND is that we can have the prediction target be deterministic (bypassing Factor 2) and by using the architecture of the target, the predictor can represent the function class of the target (bypassing Factor 3). These choices make RND immune to the noisy-TV problem.

2.4 Sequential Intrinsic Reward Generator (SRG)

RND only measures the novelty of the state, without considering the novelty of behavior. In addition, it does not support exploration in POMDP [11]. Therefore, SRG [5] has been proposed to address this issue. SRG not only measures the novelty of a certain state but extends it to measure the novelty of a sequence of states. For example, when RND transitions from $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_0$, each step will generate intrinsic rewards until s_3 is visited, but there is no intrinsic reward when s_0 is visited the next time. This is because the agent has already visited s_0. On the other hand, in SRG, the sequence when s_0 is firstly visited is s_0, but when s_0 is secondly visited, the sequence is $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_0$. The two sequences are different. It can be said that $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_0$ is a novel sequence for s_0. Therefore, even if you visit s_0 again, intrinsic rewards will be generated. Since SRG also retains the history of states, it can respond to POMDP. The intrinsic reward calculation process is shown in the Fig. 2. The intrinsic reward formulate as follows.

$$\begin{aligned} i_t = {} & \alpha\frac{1}{N_1}\|f_1(s_{t+1}) - \hat{f}_1(s_{t+1})\|^2 \\ & + (1-\alpha)\frac{1}{N_2}\|f_2(\phi(s_{t+1})) - \hat{f}_2(\phi(s_{t+1}))\|^2 \end{aligned} \tag{4}$$

where α is a hyperparameter. N_1 and N_2 are the number of output dimensions. f_1 and $\hat{f}_1$ have same architecture with RND. f_2 and $\hat{f}_2$ are newly extended DNNs and $\hat{f}_2$ includes recurrent layer.

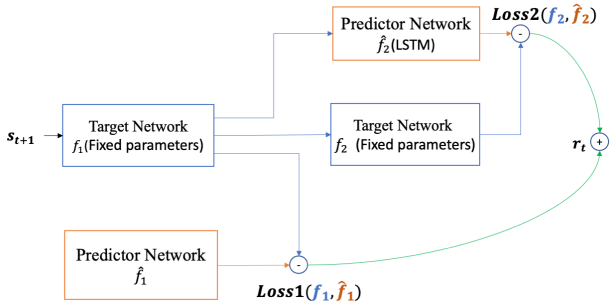

Fig. 2 Sequential Intrinsic Reward Generator (SRG)

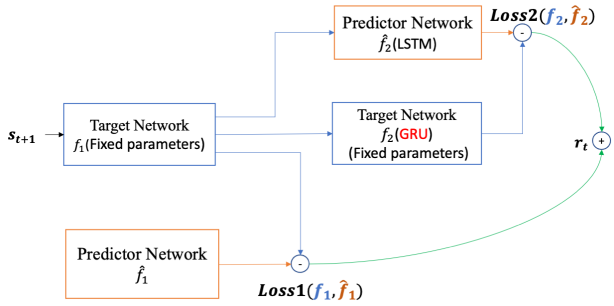

Fig. 3 Random Recurrent Network Distillation (RRND)

ϕ is a function to compress the state to a smaller size that can be fully learned. It prevents a problem that the sequence-based intrinsic rewards may be difficult to reduce by equating highly similar sequences. Due to the addition of a recurrent layer, SRG can measure the novelty of the sequence.

3. Proposal Method

RND only generates intrinsic rewards through the novelty of the state, without considering the novelty of behavior. In addition, it does not support exploration in POMDP environments. SRG has improved the problem of RND and expands the intrinsic reward to the novelty of the state sequence. However, there is a problem with SRG. In the following, first we see the problem and after that, we propose a novel method that addresses the problem.

3.1 The problem of SRG

In the previous section, we saw that SRG encourages the agent to visit new sequences. However, we found a problem that, although the predictor network in the SRG uses recurrent network to learn the features of the sequences, it still performs poorly for environments with sequential relationships. The reason is that the teacher signal provided by the target network is the features of the states rather than the features of the sequences. Suppose that we have two sequences; one is $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3$ and the other is $s_0 \rightarrow s_2 \rightarrow s_1 \rightarrow s_3$. When the agent visits s_3 through the two sequences, the features output by the predictor network should be different because the sequences are different. According to the network architecture of SRG in Fig. 2, on the other hand, the output of the target network is the feature of s_3 at this time, which the predictor network will learn. Therefore, the predictor network will gradually correct its own "mistake", so that regardless of input $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3$ or $s_0 \rightarrow s_2 \rightarrow s_1 \rightarrow s_3$, it will output the feature of s_3. That is to say, the recurrent neural network will lose the ability to extract sequence features after training and can only extract features of the current state. To improve this problem of SRG, we propose a novel type of intrinsic exploration bonus which we will introduce next section.

3.2 Generating Intrinsic Rewards by Random Recurrent Network Distillation (RRND)

We propose a novel type of intrinsic exploration bonus which will reward the agent when a new sequence is discovered. The intrinsic reward is the error of a recurrent neural network predicting features of the sequences given by a fixed randomly initialized recurrent neural network. The architecture of RRND is shown in the Fig. 3. The intrinsic reward formulate as follows.

$$\begin{aligned} i_t = \alpha \frac{1}{N_1} \| f_1(s_{t+1}) - \hat{f}_1(s_{t+1}) \|^2 \\ + (1-\alpha) \frac{1}{N_2} \| f_2(\phi(s_{t+1})) - \hat{f}_2(\phi(s_{t+1})) \|^2 \end{aligned} \tag{5}$$

where α is a hyperparameter. N_1 and N_2 are the number of output dimensions. f is the target network and $\hat{f}$ is the predictor network. f_1 and $\hat{f}_1$ have same architecture with RND. f_2 and $\hat{f}_2$ are similar to SRG but both contain recurrent networks. Because of Factor 3 of the prediction errors, the model class of predictors needs to fit the complexity of the target function. Therefore, $\hat{f}_2$ must have stronger expressive ability than the f_2. GRU [12] is similar to LSTM [13], both of which are recurrent neural networks, but the expression ability of GRU is not as good as LSTM. Thus, we add GRU layer to f_2 while $\hat{f}_2$ uses LSTM.

In addition, RRND uses various normalization to ensure that the scale of intrinsic rewards is consistent in various environments. First it normalizes the scale of rewards to alleviate the non-stationarity of reward functions. Second, it normalizes the observations to ensure that they have less variation across different environments. Finally, RRND uses feature normalization, which is shown as useful by Burda [3], to ensure that the scale of intrinsic rewards is consistent throughout the state space. We use this method in our work.

The pseudo-code of RRND is shown in Algorithm 1.

4. Experiment

In this section, we introduce the detail of the experiment, the optimization objective, the hyperparameters, and the architecture of the neural network used in this experiment. In all this experiment, we used a machine having two Intel Xeon E5-2650 v4 CPUs and one NVIDIA GeForce GTX1080 Ti. Tensorflow 1.5, OpenAI Baselines, CUDA, and cuDNN were used as calculation libraries.

4.1 Environment

We used OpenAI Gym's Atari2600 [14] as benchmark to test the performance of the RRND algorithm. We chose three games used in the SRG paper: Montezuma Revenge, Private Eye and PitFall! These are called hard exploration environments because

Algorithm 1 RRND pseudo-code

$N \leftarrow$number of rollout
$K \leftarrow$length of rollout
$c_e, c_i \leftarrow$coefficient of extrinsic and intrinsic reward
$\gamma_e, \gamma_i \leftarrow$ discount rate of extrinsic and intrinsic reward
$t = 0$
Sample state $s_0 \sim p_0(s_0)$
while $z = 0$ to N **do**
  while $j = 1$ to K **do**
    Sample action $a_t \sim \pi(a_t|s_t)$
    Sample state s_{t+1}, extrinsic reward e_t, terminal $d_t \sim p(s_{t+1}, e_t|s_t, a_t)$
    Calculate intrinsic reward
      $i_t = \alpha\|\hat{f}_1(s_{t+1}) - f_1(s_{t+1})\|^2 + (1-\alpha)\|\hat{f}_2(s_{t+1}) - f_2(s_{t+1})\|^2$
    Calculate state value $v_{i,t}$ and $v_{e,t}$
    Add $s_t, s_{t+1}, a_t, e_t, i_t, d_t, v_{i,t}, v_{e,t}$ to batch B_z
    $t+ = 1$
  end while
  Normalize the intrinsic rewards contained in B_z
  Normalize the extrinsic rewards contained in B_z
  Calculate returns $R_{i,z}$ using γ_i and intrinsic reward
  Calculate advantages $A_{i,z}$ using $R_{i,z}$ and state value $V_{i,z}$
  Calculate returns $R_{e,z}$ using γ_e and extrinsic reward
  Calculate advantages $A_{e,z}$ using $R_{e,z}$ and state value $V_{e,z}$
  Calculate combined advantages $A_z = c_i A_{i,z} + c_e A_{e,z}$
  Update observation normalization parameters using B_z
  Train policy network parameters on batch $B_z, R_{e,z}, R_{i,z}, A_z$ using Adam
  Update feature normalization parameters using B_z
  Train RRND parameters on batch B_z using Adam
end while

it is difficult to obtain extrinsic rewards. In this experiment, we studied the difference between RRND, SRG and RND. The state that the agent can acquire is the images of game screen. These images are called observations. The agent will get +1 extrinsic reward when it gets the score. When the agent loses score, it will get −1 extrinsic reward.

Here, we detail the key setting we made in the all experiment. These settings were to reduce non-stationarity in order to make learning more stable and consistent in different environments.

(1) *Reward normalization.* We normalized the scale of rewards by dividing the rewards by a running estimate of the standard deviation of the sum of discounted rewards.
(2) *Feature normalization.* We use batch-normalization algorithm [15] to achieve this.
(3) *Observation normalization. Steps of rollout* means the step length of one simulation. We run a 50 times *steps of rollout* steps random agent on the target environment, and then calculate the mean and standard deviation of the observations, and use them to normalize the observations during training.

4.2 Architecture of Neural Networks

RRND is a DNN used to output intrinsic rewards. The architecture of f_1 and $\hat{f}_1$ are same as RND. The architecture of $\hat{f}_2$ is determined by preliminary experiments.

(1) Increase the fully-connected layer in $\hat{f}_2$.
(2) Test in Montezuma Revenge.
(3) Check the intrinsic rewards. If the decrease in intrinsic rewards is faster than the previous architecture, return to 1, and if it is slower, the previous architecture is adopted.

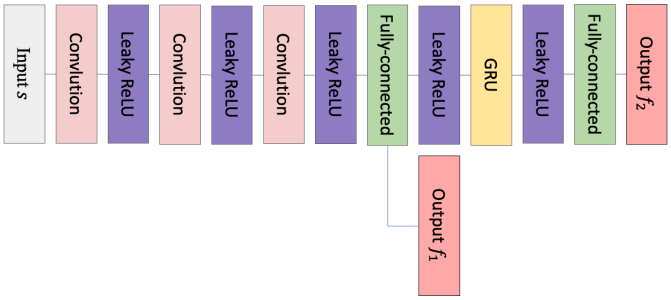

Fig. 4 Architecture of Target Network

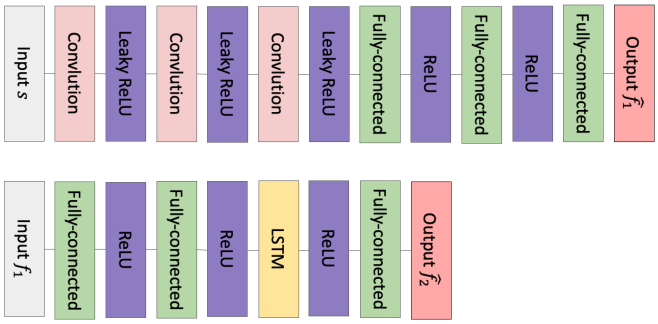

Fig. 5 Architecture of Predictor Network

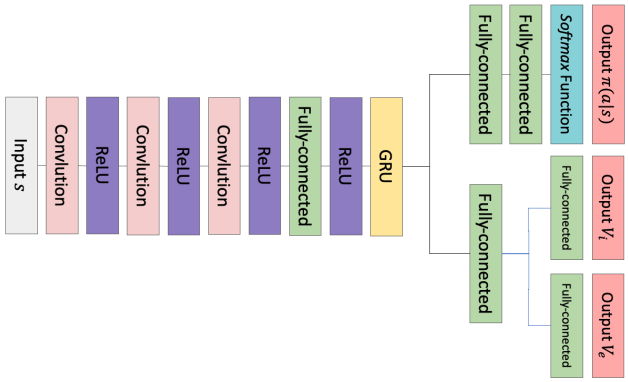

Fig. 6 Architecture of Policy Network.

The policy network is a DNN used to output $\pi(a|s)$, V_e and V_i. $\pi(a|s)$ is the probability distribution of actions. V_e is a value function based on extrinsic rewards, and V_i is a value function based on intrinsic rewards. The architecture of neural network in this experiment is shown in Figs. 4, 5 and 6.

4.3 Optimization Problem

The error function of intrinsic rewards is defined as follows.

$$L_{intrinsic_{f_1}}(\theta_{Int1}) = \frac{1}{M}\sum_{i=0}^{M}\|f_{1i} - \hat{f}_{1i}\|^2 \tag{6}$$

$$L_{intrinsic_{f_2}}(\theta_{Int2}) = \frac{1}{M}\sum_{i=0}^{M}\|f_{2i} - \hat{f}_{2i}\|^2 \tag{7}$$

where M is the number of data used to update the parameters.

The overall optimization problem that is solved for training the agent is

$$\min_{\theta_\pi,\theta_{Int1},\theta_{Int2}} \left[L_{PPO}(\theta_\pi) + \alpha L_{intrinsic_{f_1}}(\theta_{Int1}) + (1-\alpha) L_{intrinsic_{f_2}}(\theta_{Int2}) \right] \tag{8}$$

where θ_π are the parameters of the policy network. α is same as equation 5. Same as RND and SRG, we used PPO [16] as learning method for the policy network. $L_{PPO}(\theta_\pi)$ is the loss function in PPO algorithm. θ_{Int1} and θ_{Int2} are the parameters of the neural networks that generate intrinsic rewards. These parameters are updated by optimization algorithm Adam [17].

4.4 Hyperparameters

The hyperparameters are shown in Table 1. We set the coefficient of extrinsic reward higher than intrinsic reward because we do not want the agent to focus solely on obtaining intrinsic rewards. If α is set to 0, the agent may spend a lot of time to explore useless sequences. This is inefficient. To prevent this, we set α to 0.5. We use the episode length for the evaluate length of sequence, because it is more likely to find a correct sequence to complete the game from various sequences as the length of the sequence is longer. If γ_E higher than γ_I, the agent will focus on extrinsic rewards. Clip range is a hyperparameter in PPO. We used the same scale as PPO.

Table 1 Hyperparameters: $c_e, c_i, \gamma_e, \gamma_i$ and rollout length are in Algorithm 1, α is in Equation 5.

Hyperparameter	Value
Number of parallel environments	32
Rollout length	128
c_e	2
c_i	1
γ_E	0.99
γ_I	0.99
α	0.5
Clip range	[0.9, 1.1]
Learning rate of policy network	0.0001
Learning rate of $\hat{f}_1$	0.0001
Learning rate of $\hat{f}_2$	0.0001
Evaluate length of sequence	length of episode

5. Results and Discussion

In this section, we will discuss the experimental results based on the experimental design and details in Section 4.

5.1 Montezuma Revenge

The results of Montezuma Revenge are shown in the Figs. 7 and 8. We can find that RRND performs better than SRG and RND in both the average episodic score and the average number of rooms found. In addition, our method can reach a higher score faster. For example, the average episodic score of RRND has exceeded 4,000 at about 7,000 parameter updates, but both SRG and RND took more than 10,000 parameter updates to get the score. In all our experiments, the best one of these runs was that 21 rooms had been found in 35,000 parameter updates. Due to the limitation of computing resources, we could not continue the experiment. Finally, when we watched the video of the agent playing Montezuma Revenge, we found that the agent did learn sequential relationship. For example, when the agent found a key, they would actively look for a door, no matter where the key was found. However, unfortunately, this may reduce the efficiency of exploration, because sometimes the agent will return to the place where the door has been opened.

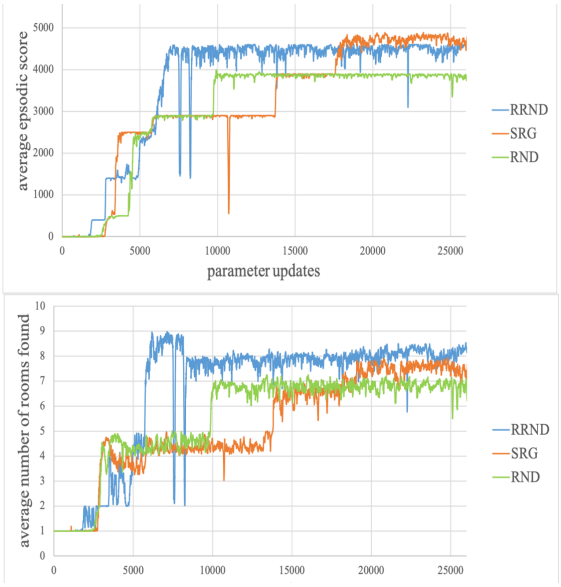

Fig. 7 The performance in Montezuma Revenge. The upon shows the average episodic score and the below shows the average number of rooms found. The blue curve represents RRND, the green curve represents RND, and the orange curve represents SRG.

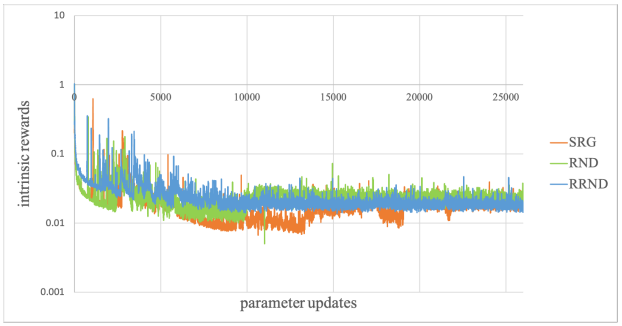

Fig. 8 The intrinsic rewards in Montezuma Revenge. The blue curve represents RRND, the green curve represents RND, and the orange curve represents SRG.

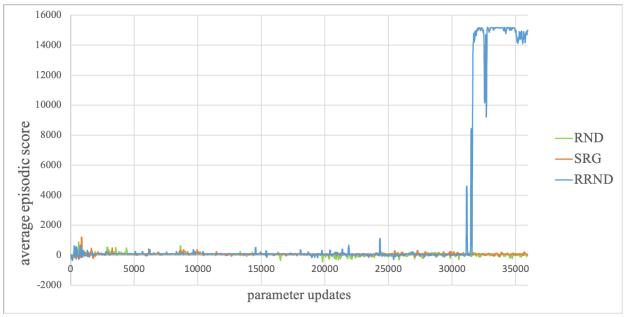

Fig. 9 The performance in Private Eye. The blue curve represents RRND, the green curve represents RND, and the orange curve represents SRG.

5.2 Private Eye

The results of Private Eye are shown in the Figs. 9 and 10. We found that the RRND algorithm performed very well in this game. After updating about 30,000 times, the agent scored from 5,000 to 9,000 in a short time, and finally reached 15,000. Com-

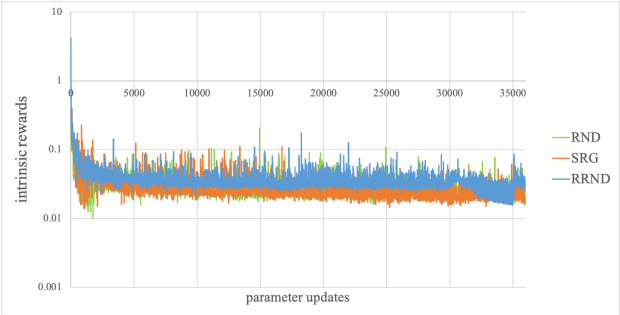

Fig. 10 The intrinsic rewards in Private Eye. The blue curve represents RRND, the green curve represents RND, and the orange curve represents SRG.

pared with RRND methods, both of RND and SRG have almost no score. In Private Eye, if you want to get a higher score, you must get a specific item and return to the corresponding room. So in some cases, you have to go back to the room you have visited (this is a common situation in this game). However, agents who use the RND algorithm to explore, in theory, will not return to the original room because returning to the room they have visited will not generate intrinsic rewards. Similarly, due to the problem mentioned above, that is, the recurrent network has not learned the sequential feature, SRG also will not return to the room that have visited. Thus, both RND and SRG are difficult to get a high score. On the other hand, RRND generates intrinsic rewards according to different state sequences. For RRND, returning to the visited rooms is a different order, which means that returning to the visited rooms will generate intrinsic rewards. Therefore, the agent has the motivation to return to the room that has been visited after obtaining the item.

6. Conclusion

In this work, we propose a novel type of intrinsic exploration bonus which will reward the agent when a new sequence is discovered. The intrinsic reward is the error of a recurrent neural network predicting features of the sequences given by a fixed randomly initialized recurrent neural network. We tested the proposed method called RRND in the hard exploration environment of Atari2600 and found that RRND performed very well in environments where the sequential relationship is very important, such as Private Eye. At the same time, the performance in other hard exploration environments is better than the other two methods. Therefore, RRND does improve the problem of recurrent neural network degradation in the SRG method.

However, on the other hand, we also found a limitation with RRND. Suppose the environment shown in the Fig. 11. When using RRND or the others, such as SRG, RND, etc., since some area on the left have been explored by the agent, the agent will choose to explore the right, but the real goal is on the left. This causes the agent to fail to find the goal. It is a future work to address this limitation. We can possibly use the Intra-life intrinsic rewards [18] to solve the problem because they encourage the agent to periodically revisit familiar (but probably not fully explored) states over several episodes.

Acknowledgments This work was partly supported by JSPS KAKENHI Grant Number JP19K12118.

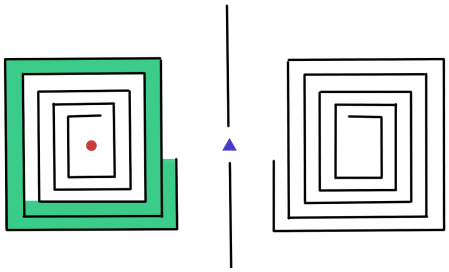

Fig. 11 Example of an environments where RRND will probably fail to arrive the goal. Green represents the area that has been explored, while red represents the goal and blue represents the agent.

References

[1] D Zhao, K Shao, Y Zhu, D Li, Y Chen, H Wang, D Liu, T Zhou, C Wang. Review of deep reinforcement learning and discussions on the development of computer Go, Control Theory and Applications 33 (6), 701-717, 2016

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Belle-mare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen,Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra,Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning.Nature, 518(7540):529–533, February 2015.

[3] Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. Exploration by random net-work distillation. In 7th International Conference on Learning Representations, ICLR 2019, NewOrleans, LA, USA, May 6-9, 2019.

[4] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In proceedings of the IEEE Conference on Computer Vision andPattern Recognition Workshops, pp. 16–17, 2017.

[5] Kazuhiro Murakami, Koichi Moriyama, Tohgoroh Matsui, Atsuko Mutoh, and Nobuhiro Inuzuka. Exploration Improvement by Sequential Intrinsic Reward Generator in Deep Reinforcement Learning. IPSJ Transactions on Mathematical Modeling and Its Applications, 14(1):1-11, 2021. (in Japanese).

[6] E. L. Ryan, Richard; Deci. Intrinsic and extrinsic motivations: Classic definitions and new directions.Contemporary Educational Psychology, 2000.

[7] R. W. White. Motivation reconsidered: The concept of competence. Psychological Review,66:297–333, 1959.

[8] Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches.Frontiers in neurorobotics, 1:6, 2009.

[9] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models.arXiv preprint arXiv:1507.00814, 2015.

[10] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A. Efros. Large-scale study of curiosity-driven learning. In arXiv:1808.04355, 2018.

[11] Åström, K.J. (1965). Optimal control of Markov processes with incomplete state information. Journal of Mathematical Analysis and Applications. 10: 174–205.

[12] Junyoung Chung et al.: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, arXiv: 1412.3555 [cs.NE] (2014)

[13] Sepp Hochreiter, Jurgen Schmidhuber.: Long short-term memory, Neural Computation, 9(8), p. 1735–1780 (1997)

[14] Brockman, Greg, Cheung, Vicki, Pettersson, Ludwig, Schneider, Jonas, Schulman, John, Tang, Jie, and Zaremba, Wojciech.: Openai gym. arXiv:1606.01540 (2016).

[15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift.arXiv preprint arXiv:1502.03167, 2015.

[16] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

[17] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. ICLR, 2015.

[18] Christopher Stanton and Jeff Clune. Deep curiosity search: Intra-life exploration can improve performance on challenging deep reinforcement learning problems.arXiv preprint arXiv:1806.00553,2018.