IEICE Communications Express, Vol.12, No.11, 575–578

ComEX

LETTER

Implementation and evaluation of NoisyNets to reinforcement learning of automated ICT system design

Tianchen Zhou[1, a)], **Yutaka Yakuwa**[2, b)], **Natsuki Okamura**[1, c)], **Takayuki Kuroda**[2, d)], **and Ikuko Eguchi Yairi**[1, e)]

Abstract This paper proposes to apply the method with an additional noisy layer to the structure of the graph neural network for reinforcement learning in automated design technology for information and communication systems. The automated design technology has an elementary problem of huge learning time caused by overestimating a specific configuration because of hardly ever rewards despite huge exploration space with a vast combination of selections, arrangements, and connections. The parametric noise applied to the network is learned with gradient descent and the remaining network weights to reduce this harmful overestimation during learning and increase the design exploration efficiency. The evaluation result showed that using the proposed algorithm for our automated design technology in development could shorten 15% of the episodes needed for learning to converge.

Keywords: network system design, design automation, machine learning, reinforcement learning

Classification: Network management/operation

1. Introduction

Design of an ICT system providing certain network application services in a wide area, which consists of selecting appropriate equipment for various requirements, optimal arrangements, and correct connections, needs specialized knowledge and enormous human power. Recently, deep reinforcement learning (DRL) has provided a powerful design framework for design in distinguished fields [1], such as drug synthesis [2], energy systems [3] and semiconductor chip design [4]. Automating some parts of the ICT system design process has been proposed by applying DRL [1]. Application of reinforcement learning usually faces the challenge of environments containing vast combinations but sparse rewards which leads to a problem of huge learning time [5]. Automated ICT system design technology using DRL faces the same problem due to the sparse rewards despite a vast combination of selections, arrangements, and connections in the ICT system, which should be solved. The speed for updates on ICT system products is fast [6]. Shortening the learning time is crucial for ensuring that the automated design technology can keep up with the frequent updates on the information of ICT system products which is necessary for performing auto-design in industrial operations. The problem should be solved by improving the efficiency of learning algorithms that function quickly and flexibly for obtaining more optimal design results beyond human ability with automated ICT system design [7]. The cause of this problem is that very few situations in search obtain rewards from the vast combination of ICT system design and some random reward obtained accidentally will likely be overestimated and cause a learning efficiency slowdown. As proposed solutions for this problem, the application of Heuristic Search [8] and Double DQN [7] have achieved a 20%-25% increase in learning efficiency. The design of an ICT system can be denoted by graphs with nodes as objects and edges representing the relationships between adjacent nodes [9]. The automated ICT system design technology in this paper uses Graph Neural Network (GNN) to calculate each state value for the system layout represented by graphs. This paper challenges to adapt NoisyNets, which was applied to the typical neural network (NN) in the original article [10], to the GNN of automated ICT system design technology.

2. Methodology

2.1 Weaver

Weaver is an exploration based automated ICT system design technology originated by NEC [8]. The overview of Weaver's architecture is shown in Fig. 1(i). The essential elements are a) system requirement (SR) as the input, b) concrete system configuration (CSC) as the output, c) search tree carrying reification from SR to CSC, d) reification rules (RR) applied to the search tree, and e) DRL model of search. SR are shown as directed graphs called topologies representing the components and their relationships as nodes and directed edges, an example shown in Fig. 1(ii). Parts with solid lines indicate the details or relationship are defined as "concreted," and features with dot lines indicate they are "still abstract." Figure 1(iii) is an example of reification in that the abstract part representing "application and database HTTP connection" is converted to "TCP connection between each running VM." Figure 1(iv) shows an example of the search tree and applicable RR. Weaver converts abstract parts in a topology (a) into more concrete figuration (b)-(d) and (e) step by step based on the applied RR (X)-(Z) from many different RRs. The search ends ei-

[1] Graduate School of Science and Technology, Sophia University, 7–1 Kioi-cho, Chiyoda-ku, Tokyo, 102–8554, Japan
[2] NEC Corporation, 5–7–1 Shiba, Minato-ku, Tokyo, Japan
[a)] tianchensp@yairilab.net
[b)] y-yakuwa@nec.com
[c)] okamura@yairilab.net
[d)] kuroda@nec.com
[e)] i.e.yairi@sophia.ac.jp

DOI: 10.23919/comex.2023XBL0101
Received July 19, 2023
Accepted August 3, 2023
Publicized October 17, 2023
Copyedited November 1, 2023

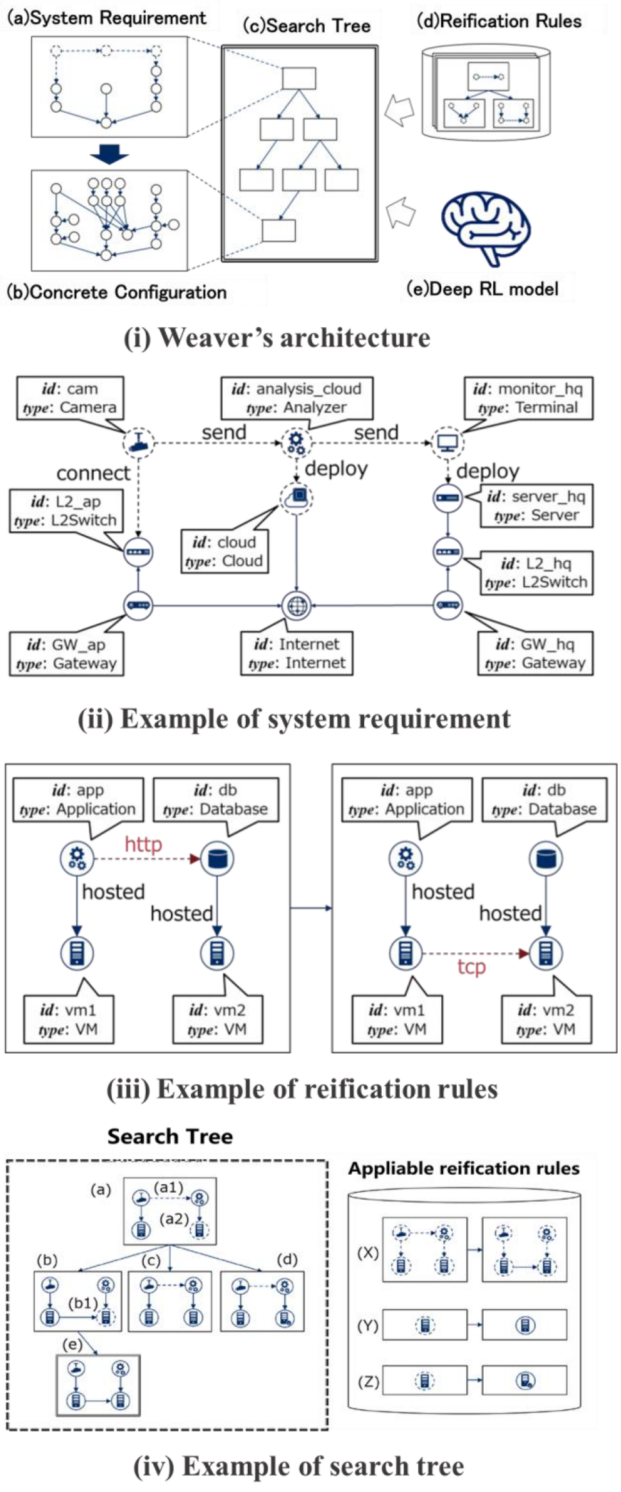

Fig. 1 (i) Overview of Weaver's architecture. (ii) An example of system requirement. (iii) An example of reification rules. (iv) An example of search tree

ther when all abstract elements are embodied, or all patterns are tried. In this designing process, the DRL model chooses better topologies by estimating each topology's state value with GNN. Weaver has adopted graph nets [9] as the GNN model and DQN as the base algorithm of DRL. This paper proposes to apply NoisyNets for the GNN part of Weaver to deal with the problem of overestimating the selected action because it is hardly ever rewarded. Our previous paper used DDQN as the base algorithm of DRL instead of DQN to cope with the same problem.

2.2 NoisyNets

A typical DQN-based algorithm has the feature of choosing the action with the maximum Q value when learning [11]. As explained in the introduction there is a problem that DQN overestimates the selected action, and the possibility increases as the state space increases [5]. One of the representative methods to deal with these problems is known as NoisyNets [10], which aims to stimulate exploration and meanwhile suppress overestimation by injecting noise into the NN. The NoisyNets redefined the NN as following Eq. (1).

$$y_{noise} \stackrel{\text{def}}{=} (\mu^w + \sigma^w \odot \varepsilon^w)x + \mu^b + \sigma^b \odot \varepsilon^b. \quad (1)$$

Weight w and bias b of a typical NN are decomposed as μ^w, σ^w, μ^b, σ^b where μ and σ represent mean and variance of a normal distribution. ε^w, ε^b are randomly sampled noise parameter from a normal distribution and $\odot$ is the notation of tensor product. This method aims to mitigate the negative effect of overestimation by injecting stochastic noise ε^w and ε^b for the action selection. Noise injection as learnable parameters with backpropagation performs similarly to the self-annealing method [12]. w and b are approximated like $w \sim N(\mu^w \sigma^w)$, $b \sim N(\mu^b, \sigma^b)$, and the μ, σ of a normal distribution also can be treated as learnable parameters in the NN. The learnable parameters' initialization follows the original paper of NoisyNets. Taking the size of input as p, μ^w and μ^b are sampled from $U\left[-\frac{1}{\sqrt{p}}, \frac{1}{\sqrt{p}}\right]$. σ^w and σ^b are sampled as a constant $\frac{\sigma_0}{\sqrt{p}}$ with a hyper-parameter σ_0 (default 0.5).

2.3 Applying NoisyNets to GNN

NoisyNets was originally an idea to replace the Dense Layer (fully connected layer) of NN used in the DQN algorithm with the Noisy Layer. This paper addresses the problem of GNN overestimating a specific RR in the design process by adding a Noisy Layer to the last fully connected layer of GNN. Our previous paper confirmed the high effect of Double DQN introduced instead of DQN [7]. Combining noisy GNN and Double DQN is anticipated to be more effective in future endeavor. Figure 2 shows the image of using noisy GNN to estimate the next possible transition of topology. Topology s has attribute values N of nodes, E of edges, and u of globals. The GNN with NoisyLayer conducts a convolution for the feature and outputs an updated graph of N', E', and u' with noises. Weaver uses the V value, which represents the value of the state alone instead of the Q value [7]. V value is calculated with s as the previous state and the updated values N' and E' for maximum V value selection. Weaver uses this unique structure of NoisyGNN + DQN algorithm to perform learning. In design AI research, there is no other example of applying a Noisy Layer to GNN; this method is a new approach.

3. Evaluation and result

The experiment compares the learning convergence speed of

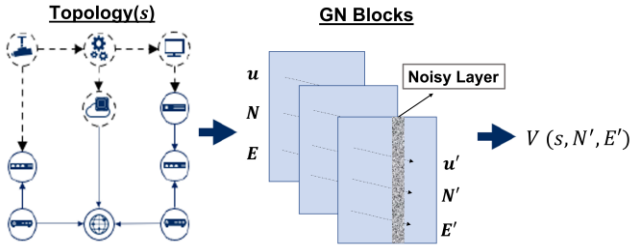

Fig. 2 The image of using noisy GNN to estimate the next possible transition of topology.

the original DQN and our proposed method of NoisyNets as follows: (1) to have a try with the testing question after every 100 episodes of learning, (2) to record the number of search steps for each try and compute the moving average across ten consecutive tries, within a single trial comprising 80 tries which are equivalent to 8,000 episodes, (3) to calculate the average of the moving average of 10 trials, and (4) to evaluate the episodes number in each algorithm when the number of search steps converges to around 50 based on the property of the testing question [8].

Weaver is currently in the developmental stage, focusing on evaluating learning efficiency, and does not yet provide a quantitative assessment of other criteria, such as design quality. In this paper, average converged episode numbers are used as the elementary evaluation metric for measuring the time efficiency of the reinforcement learning algorithm [7].

Figure 3 shows the experimental results of the convergence speed increased by about 15%. The results with NoisyNets of different initial factor values (σ_0) 0.1/0.5/1.0 are 6,400/6,300/6,500 episodes compared to 7,500 episodes for the original DQN. Referring to Fig. 3(A), setting σ_0 at a higher value gives high initial values for σ^w and σ^b, which leads to higher start values of search steps as expected. The higher value stimulates the exploration at the early stage of learning and leads to a faster convergence speed in later episodes. Even though the three cases have quite different start values, they converge at almost the same number of episodes, and the difference between them is only about 100 episodes. These differences are a slight variation because Weaver measures the search steps for the test question every 100 episodes. The convergence speed is at the top when σ_0 is set to 0.5, and hereafter the result of $\sigma_0 = 0.5$ is the primary result for applying NoisyNets. Referring to Fig. 3(B) of only the primary result and the original DQN, although the number of search steps for the NoisyNets is much higher than the original DQN in early episodes, both lines first intersect at about 300 search steps at almost the same time (around 2000 episodes). This shows that the Noisy DQN has a faster convergence speed than the original DQN, which may also be told by comparing the difference in curvature of both lines. This observation is undoubtedly considered to be the facilitation effect of perturbations from the NoisyNets. As the circled part in Fig. 3(B), the original DQN encounters stagnation at (1) 3,000-4,000 episodes and (2) 5,500-6,500 episodes because of the negative effect of overestimation. At (1), a similar stagnation was observed in the NoisyNets but considerably eased. The stagnation at (2) is entirely resolved with the NoisyNets. These results prove that NoisyNets is

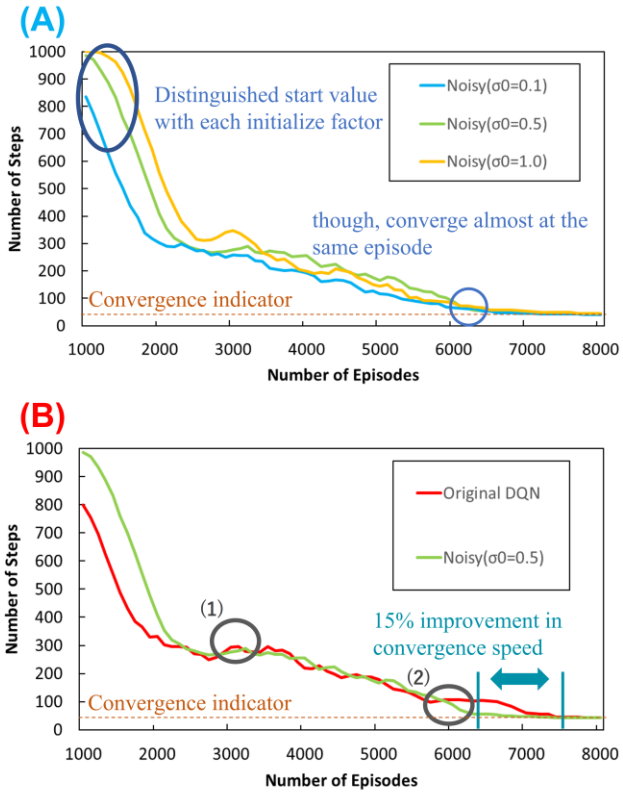

Fig. 3 (A) Results for NoisyNets. (B) Results comparing with original DQN

Table I Statics of original DQN and Noisy DQN

	Original DQN	Noisy DQN
Mean	7500	6300
Standard deviation	1100	400
trial	10	10

effective in mitigating overestimation.

To further evaluate the significance of our proposed Noisy-DQN compared to the original DQN, we perform a simple 2-sample t-test, which is considered to be a popular significance measure metric for reinforcement learning [13], on the average convergence episodes of two. The statistics for the t-test are provided in the following Table 1. The difference between original DQN and Noisy DQN is statistically significant ($t(18) = 3.24$, $p = 0.008$) in a significance level of $\alpha = 0.01$.

4. Conclusion

This paper verified the effectiveness of introducing NoisyNets to the GNN of the automated ICT system design technology Weaver. The number of episodes for convergence indicates a valid improvement in learning efficiency by about 15%. NoisyNets successfully suppresses harmful overestimation done by GNN before value prediction and action selection by DQN. Our previous research used Double DQN without NoisyNets on GNN, and this method worked to decline overestimation too [7]. As expected, these two methods can be easily combined with no conflict for further improvement. A further task is in progress for applying or integrating the remaining methods of Rainbow other than

NoisyNets and Double DQN. In the future, we would like to realize a full-force Rainbow agent [14] in Weaver and intend to explore methods for measuring the quality of the output design.

References

[1] H. Sun, H.V. Burton, and H. Huang, "Machine learning applications for building structural design and performance assessment: State-of-the-art review," *Journal of Building Engineering*, vol. 33, Jan. 2021. DOI: 10.1016/j.jobe.2020.101816

[2] M.H.S. Segler, M. Preuss, and M.P. Waller, "Planning chemical syntheses with deep neural networks and symbolic AI," *Nature*, vol. 555, pp. 604–610, March 2018. DOI: 10.1038/nature25978

[3] A.T.D. Perera, P.U. Wickramasinghe, V.M. Nik, and J.-L. Scartezzini, "Introducing reinforcement learning to the energy system design process," *Applied Energy*, vol. 262, 114580, March 2020. DOI: 10.1016/j.apenergy.2020.114580

[4] A. Mirhoseini, A. Goldie, M. Yazgan, J. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, S. Bae, A. Nazi, J. Pak, A. Tong, K. Srinivasa, W. Hang, E. Tuncer, A. Babu, Q.V. Le, J. Laudon, R. Ho, R. Carpenter, and J. Dean, "Chip placement with deep reinforcement learning," arXiv preprint arXiv:2004.10746, April 2020. DOI: 10.48550/arXiv.2004.10746

[5] S. Thrun and A. Schwartz, "Issues in using function approximation for reinforcement learning," Proceedings of the 1993 Connectionist Models Summer School, Dec. 1993.

[6] M. Nakamura, "Study on ICT system today and future for engineering chain connection in manufacturing from product design to production," *Journal of the Japan Society of Presice Engineering*, vol. 81 no. 3, pp. 220–224, 2015. DOI: 10.2493/jjspe.81.220

[7] N. Okamura, Y. Yakuwa, T. Kuroda, and I.E. Yairi, "Applying double DQN to reinforcement learning of automated designing ICT system," *IEICE Commun. Express*, 2022, vol. 11, no. 10, pp. 667–672, Oct. 2022. DOI: 10.1587/comex.2022XBL0100

[8] T. Kuroda, T. Kuwahara, T. Maruyama, K. Satoda, H. Shimonishi, T. Osaki, and K. Matsuda, "Weaver: a novel configuration designer for IT/NW services in heterogeneous environments," 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, pp. 1–6, Dec. 2019. DOI: 10.1109/globecom38437.2019.9014133

[9] P.W. Battaglia, J.B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, "Relational inductive biases, deep learning, and graph net-works," arXiv preprint arXiv:1806.01261, 2018. DOI: 10.48550/arXiv.1806.01261

[10] M. Fortunato, M.G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg, "Noisy networks for exploration," arXiv preprint arXiv:1706.10295, 2017. DOI: 10.48550/arXiv.1706.10295

[11] 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, pp. 529–533, 2015. DOI: 10.1038/nature14236

[12] J. Branke, S. Meisel, and C. Schmidt, "Simulated annealing in the presence of noise," *J. Heuristics*, vol. 14, pp. 627–654, 2008. DOI: 10.1007/s10732-007-9058-7

[13] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," Proc. 32nd AAAI Conf. Artif. Intell., pp. 3207–3214, 2018. DOI: 10.1609/aaai.v32i1.11694

[14] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, no. 1, 2018. DOI: 10.1609/aaai.v32i1.11796