Progress in Nuclear Energy 118 (2020) 103123

Contents lists available at ScienceDirect

Progress in Nuclear Energy

journal homepage: www.elsevier.com/locate/pnucene

Providing support to operators for monitoring safety functions using reinforcement learning

JaeKwan Park*, TaekKyu Kim, SeungHwan Seong

Korea Atomic Energy Research Institute, 111, Daedeok-daero 989 Beon-gil, Yuseong-gu, Daejeon, Republic of Korea

ARTICLE INFO

Keywords:
Reinforcement learning
Deep neural network
Operator support
Diagnosis
Safety function status check

ABSTRACT

Continuous monitoring and diagnosis are important for safe operation of nuclear facilities. In an emergency shutdown, the diagnostic tasks can be challenging for human operators who may be under intense stress and/or lack training. In recent years, studies using artificial-intelligence technologies have been actively conducted to help in diagnostic tasks. The current study proposes a data-driven approach that leverages deep reinforcement-learning techniques to intelligently learn effective strategies for state diagnosis of safety functions. First, a learning framework and key elements of reinforcement learning are designed as basic components. Then, a deep neural-network structure and a deep reinforcement algorithm are presented for diagnosis learning. The experimental results demonstrate the feasibility of deep reinforcement learning on diagnosing the safety functions of a nuclear facility.

1. Introduction

Monitoring and diagnosing of problems by human operators are significantly associated with safe operations of nuclear facilities. State-diagnosing and decision-making tasks are especially critical for human operators to correctly perform in case of an emergency shutdown. When a research reactor is scrammed, the operators sequentially identify the reactor state and respond to abnormal conditions according to the emergency operating procedures (EOP). A safety-function status check (SFSC), defined as a section in EOP, is one of the most important tasks to perform for reactor safety. This task ensures that the safety functions of the reactor are maintained within their safety limits.

Although a procedure is provided, event diagnosis after a reactor scram is a challenging task for operators. In addition, diagnostic activities can require a considerable amount of time to accurately identify the reactor condition because the operators should analyze the trends of many parameters and alarms. Therefore, quickly recognizing the reactor transient is not easy for operators. To reduce this burden on the operators, many studies have been conducted to help them diagnose or detect accidents in nuclear plants.

Artificial neural networks (Rumelhart and McClelland, 1986) have been considered as a particularly suitable approach because a diagnostic task is similar to a pattern-recognition problem. Thus, several studies have applied artificial neural networks to diagnostic tasks (Fantoni and Mazzola, 1996; Lee and Seong, 2005; Mo et al., 2007; Santosh et al., 2007). In recent years, as hardware and software have evolved, machine learning that uses a deep neural network (Bengio, 2009; Hinton and Salakhutdinov, 2006; Krizhevsky et al., 2012) has been introduced, and related studies based on supervised learning have been published. A support system for decision-making during a severe-accident situation has been proposed for nuclear power plants (Yoo et al., 2018). The system uses a fuzzy neural network (Buckley and Hayashi, 1994) to diagnose an accident transient, which can contribute to improving the safety of a nuclear power plant by predicting an accident scenario. Yang and Kim (2018) proposed an accident-diagnosis algorithm using long short-term memory (LSTM), which demonstrated that LSTM, a type of recurrent neural network, performs well in analyzing time-series data.

Similar to the studies based on supervised learning, reinforcement-learning studies with significantly improved performance have been published in the information-technology fields. Mnih et al. (2015) introduced deep reinforcement learning (DRL), including a deep Q-network (DQN), that combined Q-learning with a deep convolutional neural network specialized for processing spatial data arrays such as images. DQN demonstrated great performance in playing Atari (Mnih et al., 2015) and Go games (Silver et al., 2016); thus, it can be considered as a powerful data-driven method. This technique overcomes the conventional reinforcement-learning problem, which was not completely observed because of the very large state space existing in a real environment, by building a deep neural network to relate the value

* Corresponding author.
E-mail addresses: jkpark183@kaeri.re.kr (J. Park), taekkyukim@kaeri.re.kr (T. Kim), shseong@kaeri.re.kr (S. Seong).

https://doi.org/10.1016/j.pnucene.2019.103123
Received 27 February 2019; Received in revised form 29 July 2019; Accepted 5 August 2019
Available online 13 August 2019
0149-1970/

List of acronyms

HVAC	Heating, ventilation, and air conditioning
ANN	Artificial neural network
KAERI	Korea atomic energy research institute
DRL	Deep reinforcement learning
LSTM	Long short-term memory
DQN	Deep Q-network
ReLU	Rectified linear unit
EOP	Emergency operating procedures
SFSC	Safety-function status check

estimates and associated state-action pairs. Wei et al. (2017) proposed a reinforcement-learning algorithm for building HVAC control based on the DRL introduced by Mnih et al. (2015). Implementation of the algorithm resulted in energy-cost reduction compared with the existing method while maintaining the room temperature within a desired range.

The present study investigates the feasibility of applying the DRL technique to continuously perform an SFSC task. First, we introduce a DRL framework and the key elements of reinforcement learning as basic components. We then develop a DRL algorithm using value approximation based on the deep neural network. The experimental results demonstrate that the DRL-based diagnosis can identify the status changes of a reactor. The remainder of this paper is organized as follows. Section 2 introduces a reinforcement-learning design for the SFSC. Section 3 presents a deep neural network for value-function approximation of the reinforcement learning. Section 4 provides the experimental results, and Section 5 concludes this paper.

2. Reinforcement-learning design for safety-function monitoring

2.1. Reinforcement-learning framework

This study develops a DRL-based SFSC framework composed of SFSC-environment and DRL modules. The SFSC-environment module (i.e., emulator) loads the sensor-signal values associated with the safety functions from dataset files and displays the safety-function status and signal values as environment variables. It also performs the action initiated by the DRL module and updates the environment variables. The DRL module defines the state, action, and reward as key elements of reinforcement learning for this application. It implements the reinforcement-learning algorithm using a deep neural network to learn the best action for each environment condition. The deep neural network, a multi-layered network structure, is used to approximate the expected reward for the input state. During training, the parameters of the nodes inside the deep neural network are gradually optimized.

The overall operation of the SFSC framework is shown in Fig. 1. First, the current signal values are introduced to the SFSC environment, and a pair of actions is output by the neural network, which is called predict Q-network. Once the action is executed in its current state, it is evaluated to determine the reward for "good" or "bad" result. These trial-and-error results are stored in the replay buffer, and the predict Q-network is periodically trained using the experience information. During the training, it learns an effective condition-diagnosis policy without using any pre-defined model. It utilizes another network, denoted as $\hat{Q}$-network, which has the same structure as the Q-network, to realize less fluctuation and more stability in the training process.

2.2. Elements of reinforcement learning

As stated in the recent safety-analysis report (KAERI, 2014), Korean research reactors (e.g., Kijang Research Reactor) feature the following safety functions. First, the reactivity-control safety function ensures that a reactor is shut down, which is of utmost importance because it provides primary means of controlling the reactivity in the reactor core. Second, maintenance of the auxiliary safety functions ensures that electrical power is successfully transferred from off-site to the internal class 1E power bus. The equipment used for the safety functions is powered by the maintenance of auxiliary safety functions. Third, the reactor-pool inventory control safety function ensures that the reactor core is covered with primary coolant to enable removal of the decay heat. It includes monitoring whether the coolant in the core is properly maintained. Next, the core heat-removal safety function ensures that heat from the reactor core is properly removed by forced convection or natural circulation. Finally, the confinement-isolation safety function ensures that confinement atmospheric conditions are maintained within acceptable limits or actions are initiated to mitigate the effects of events

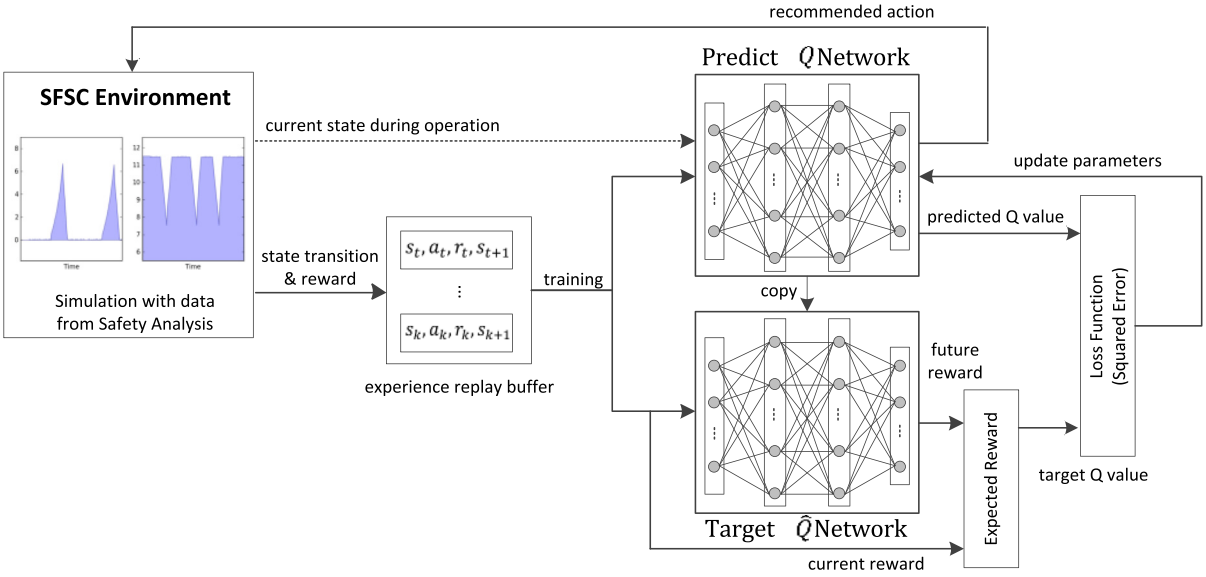


Fig. 1. SFSC framework based on reinforcement learning.

related to radiation hazards.

This work, which is presented as a feasibility study, simplifies the target environment to reduce the problem complexity. First, we assume that only three safety functions are available: reactivity control, reactor-pool inventory control, and confinement isolation. We assume that each safety function is totally affected by a critical signal such as the neutron log rate for reactivity control, reactor-pool level for reactor-pool inventory control, or pool-surface radiation for confinement isolation.

To apply the reinforcement-learning technology to this target environment, key elements for the environment need to be defined. Reinforcement learning is a learning method that involves mapping conditions into actions to maximize the numerical reward. The learning agent is not told what actions to take but instead performs a trial-and-error search to discover which actions yield the highest reward. The agent senses its environment state and performs an action that affects the state. Thus, the agent goal, environment states, possible agent actions, and rewards resulting from the actions in the target environment should be clearly defined.

Similar to human operators, the DRL needs to use immediate past and present signal values and varying patterns to analyze abnormalities associated with the safety functions. In particular, to cope with a rapid change in the state, immediately identifying a signal-increase pattern is important by learning the signals from the previous multi-steps together (instead of the simple current signal values). In other words, a sequence of signals is defined as a state in this environment. The signals in the state are encoded in a two-dimensional array, which is the input form of the DRL through a pre-processing step.

The DRL goal is to recommend an appropriate response action before the safety functions are compromised. To learn this goal, we need to define the reward that will actually be given after the action and to predict the future reward in a given state. If the DRL performs action a_t in state s_t, the state will change to new state s_{t+1}, and the DRL will receive immediate reward r_t. The immediate reward is explained as follows. First, if the DRL does nothing in a normal situation, it is rewarded with zero (0). If it makes an appropriate decision, it receives a positive reward (1). Otherwise, if it makes a wrong decision, it receives a negative reward (-1). The DRL uses $Q(s_t, a_t)$ (Mitchell, 1997) to represent the maximum reward that can be obtained by performing action a_t in state s_t. The total expected reward in state s_t is calculated as the sum of the reward received when the state changes to next state s_{t+1} and the expected reward to be obtained by performing action a_{t+1} in state s_{t+1}. Therefore, $Q(s_t, a_t)$ can be expressed in a recursive fashion as Equation (1). In Equation (1), factor $\gamma \in [0,1]$ is a discount-rate parameter for the future reward (Mitchell, 1997) and is set as 0.99 in this study. Function max$Q$ estimates the future reward using Q-network.

$$Q(s_t, a_t) = r_t + \gamma \max Q(s_{t+1}, a_{t+1}) \quad (1)$$

Fig. 1 shows that the DRL output is an action that maximizes the expected reward in a given state. A recommendation to implement the methods to maintain each safety function is defined as actions, which are described as follows: *rod insertion* for reactivity control as the neutron log rate increases, *pool-water storage injection* for reactor-pool inventory control as the reactor-pool level decreases, *confinement isolation damper close* for confinement isolation as the pool-surface radiation increases, and *NONE* for normal state of all safety functions.

Fig. 2 shows the learning process of the DRL. As introduced by Mnih et al. (2015), two Q-networks related to action and reward are used in the DRL. The predict Q-network is used to select the best action that can obtain the maximum reward. The target $\hat{Q}$-network is used to provide a labeled reward from previously learned experience. As a first step of the learning process, the DRL observes current state s_t and outputs action a_t that yields maximum reward $Q(s_t, a_t)$ using the predict Q-network. After performing the action in state s_t, it receives immediate reward r_t and observes new state s_{t+1}. Then, the DRL calculates the expected reward during the transition from state s_{t+1} to the last state using the target $\hat{Q}$-network. Thus, the target value for the predict Q-network to learn is expressed as $r_t + \gamma \max \hat{Q}(s_{t+1}, a_{t+1})$. Finally, the parameters within the predict Q-network are updated by a loss function that calculates the difference between the predicted and target values. In other words, the predict Q-network learns by comparing the predicted reward with the actual reward that was previously obtained and tried. Reinforcement learning is known to suffer from exploitation and exploration issues in that only selected actions are attempted in the training process. The present study uses the decaying ε-greedy policy (Mnih et al., 2015), which selects actions for exploration with high probability at the beginning of the training and selects actions output by the Q-network with high probability at the end of the training. The structure of the Q-networks and the reinforcement-learning algorithm are described in detail in Sections 3.1 and 3.2, respectively.

3. DRL mechanism for safety-function monitoring

3.1. Value-function approximation

Value function represents the prediction of a future reward in a state. By considering scalability, the use of the approximation method is efficient because it requires a very large state space for all state information. The present work adopts a neural-network structure called Q-network (Mnih et al., 2015) to approximate the value function. As an

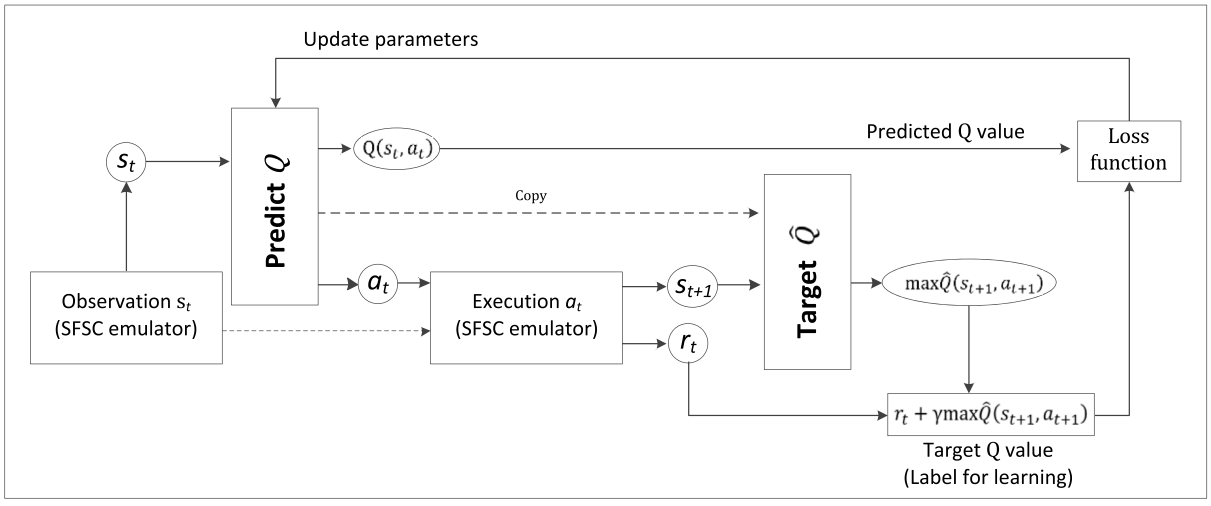


Fig. 2. Reinforcement-learning sequence.

output of the Q-network, Q-value, which is used for action selection in a certain state, can be calculated by performing a forward pass in the neural network. Fig. 3 shows the structure of the value-function approximation structure designed in this study.

The value-function approximation consists of an input processing step, deep neural-network steps (i.e., Q-network), and an output processing step. In the input processing step, the current signal values are transformed into a two-dimensional array, and this array is piled up on three arrays created in the previous execution. The prepared input is introduced to the input layer of the neural network. The layers in the neural network have 4, 32, 64, and 512 filters, and the rectified linear unit (ReLU) is used as an activation function. The last layer in the network output, namely, $Q(s_t, a_t^n)$, is an approximate value of the expected future reward by choosing action a_t^n (n is the number of action types) in state s_t. Finally, the action to obtain the maximum $Q(s_t, a_t^n)$ value is chosen as the recommended action in state s_t. During the training, the neural-network weight parameters are incrementally updated using the gradient descent algorithm with a loss function. The loss function is expressed as a squared-error function between the predicted value of the network and the labeled value. The loss function is expressed as $L = [(r_t + \gamma \max Q(s_{t+1},\ a_{t+1})) - Q(s_t,\ a_t)]^2$, where r_t is the actual current reward and γ is the discount-rate parameter, which is equal to 0.99.

This study adopts two advanced techniques (Mnih et al., 2015) for the value-function approximation: an experienced replay for a correlation problem and a network separation for a non-stationary target problem. The experienced-replay buffer shown in Fig. 1 stores records consisting of the current state, action, reward, and next state, which are the information identified from the execution of the actual action. The neural network is trained using 32 records randomly fetched from the buffer. This random sampling of the training data prevents the data-correlation problem. The network becomes non-stationary when only one neural network is used for both the predicted and target values. Then, the learning performance is known to deteriorate. As presented by Mnih et al. (2015), the current work deploys two neural networks with the same structure: one network (i.e., Q-network) is the currently trained neural network. The other one (i.e., $\hat{Q}$-network) is the network that provides a target value. Although the networks have different parameters during training, they are synchronized at every 1,000th scenario execution, as shown in Fig. 2.

3.2. DRL algorithm

The DRL learning algorithm is described in Algorithm 1. The algorithm is implemented in the SFSC emulator using datasets generated by the tool used for safety analysis of Korean research reactors. The episodes loaded in the algorithm are scenario cases extracted from the datasets, and each episode is composed of many emulation steps.

Algorithm 1
DRL algorithm for the SFSC.

1: Initialize replay buffer B = [empty]
2: Initialize predict neuralnetwork Q with weights ω
3: Initialize target neural network $\hat{Q}$ with weights $\hat{\omega} = \omega$
4: **For** episode $e = 1$ to N **do**
5: Reset the SFSC environment to the initial state
6: **For** $t = 1$ to T **do**
7: $s_t \leftarrow$ recent signal observation
8: $a_t = \begin{cases} random\ (a^1 \sim a^n) & with\ probability\ \varepsilon \\ argmax\ Q(s_t|\omega) & otherwise \end{cases}$
9: Execute action a_t in the SFSC emulator environment
10: Observe reward r_t and next state s_{t+1}
11: $B \leftarrow (s_t, a_t, r_t, s_{t+1})$
12: Get minibatch$(s, a, r, s') \leftarrow B$
13: $y = \begin{cases} r & if\ s'is\ a\ terminal\ state \\ r + \gamma max\hat{Q}(s', a'|\hat{\omega}) & otherwise \end{cases}$
14: Train $Q(\omega)$using s, a, and y by performing the gradient descent method
15: **End For**
16: Every C episode, $\hat{Q}(\hat{\omega}) \leftarrow Q(\omega)$
17: **End For**

First, the replay buffer and two neural networks are initialized. The main loop (lines 5–15) for training neural network Q is sequentially executed for each episode. The algorithm starts by observing the recent signal values and selecting an exploration action based on the ε-greedy policy or an action with a maximum value output by network Q. An information record, namely, (s_t, a_t, r_t, s_{t+1}), that observes the environment altered by the action is stored in buffer B. A batch set, which samples the records in the buffer, is prepared as input data for training network Q. In addition, the target value calculated by target network $\hat{Q}$ is prepared as the labeled data of the training. Then, the parameters inside network Q are updated as the training proceeds by performing gradient descent using the input and labeled data. After the main loop, network $\hat{Q}$ is periodically updated by copying parameters ω of network Q.

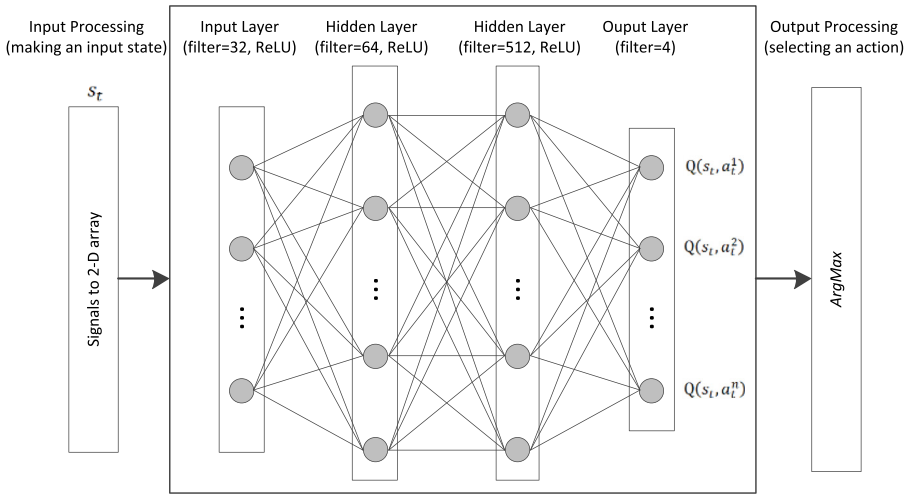
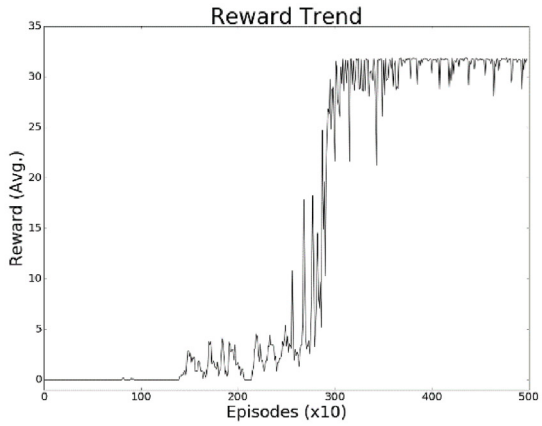
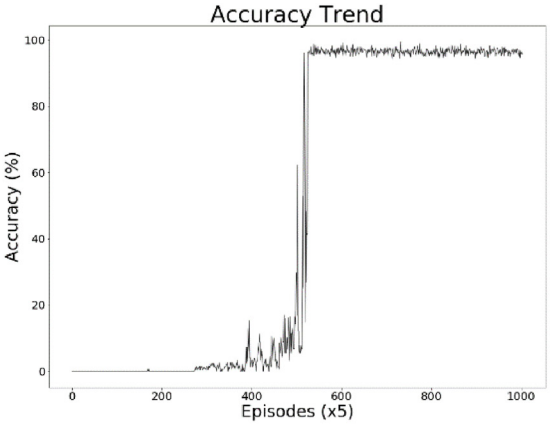


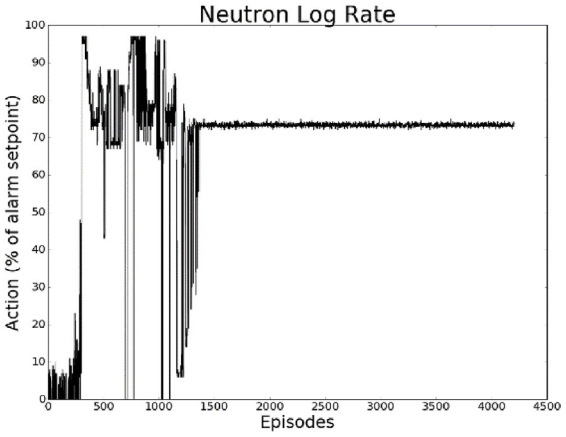
Fig. 3. Structure of the value-function approximation.



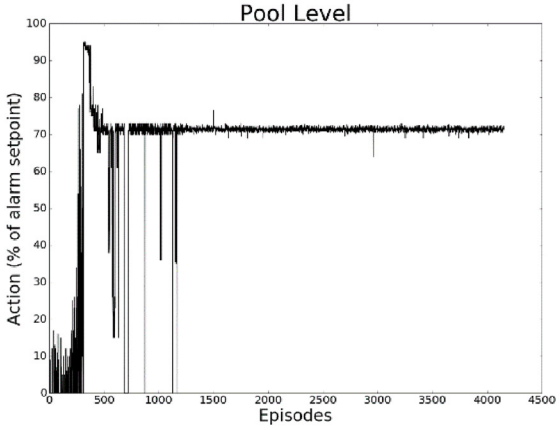
(a) Average reward



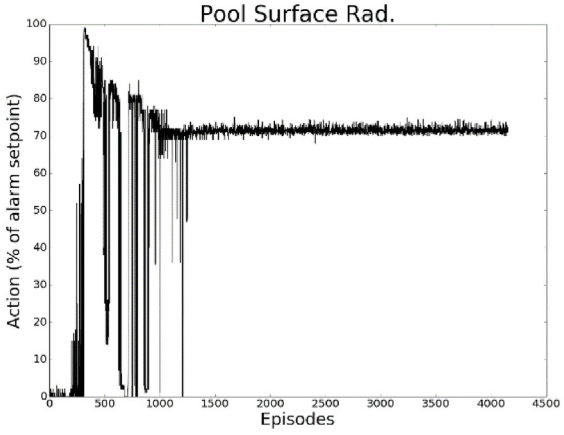
(b) Accuracy



(c) Neutron log rate



(d) Reactor-pool level



(e) Pool-surface radiation

Fig. 4. Performance of the DRL algorithm during the training process.

4. Experimental results

4.1. Experimental setup

The proposed DRL-based algorithm is implemented on an emulator for reinforcement learning, which operates using the scenario data produced from RELAP/MOD3.3 (NRC, 2001), a realistic nuclear-system transient-analysis code used for safety analysis of Korean research reactors. It is trained on the profiles of the sensor signals under excessive reactivity-insertion accidents due to the withdrawal of a control rod and loss of coolant accidents due to flow-pipe rupture. In addition, signal patterns that converge to a specific value after being increased are generated to simulate the radiation released on the reactor-pool surface. A training dataset of 10,000 episodes is set by adding ± 30% random noise to the prepared data, and a test dataset of 5,000 episodes is set by adding ± 10% random noise. The training and testing datasets are split. The free parameters of the neural networks used in the Q-network are as follows: the learning rate is 10^{-6}, discount rate is 0.99, number of mini-batches is 32, and number of hidden layers is two.

As discussed in Section 2, the state consists of three critical signals: *neutron log rate, reactor-pool level, and pool-surface radiation*. Further, the DRL algorithm selects one of the following four actions: *Insertion of rods, pool-water supply tank value open, confinement-isolation damper close, and NONE*. A positive reward is awarded if the appropriate action is recommended within the range of 70%–100% of the set point for the alarm to issue a warning that a safety function encounters a problem. Otherwise, a negative reward is awarded, and the episode is terminated. We assume that the action is performed without delay and immediately affects the state change. The maximum reward per episode is set for a rapid training process. If the total reward is greater than the maximum reward, the reward is reported, and a new episode is started.

Fig. 4 shows that the performance of the DRL gradually improves as the training progresses. This experiment selects a random signal among the target signals, proceeds through the episode, and repeats the process of experience learning. Fig. 4(a) and (b) show that the reward and accuracy are very low at the start of the training because of the penalty that results from misjudgment and wide exploration. However, they increase as the DRL algorithm learns an effective strategy for recommending an appropriate action in each signal state. Eventually, the reward stabilizes to the maximum reward, and the accuracy converges to approximately 95% after the DRL algorithm learns the strategy. Fig. 4(c) shows the learning process to offer an appropriate action in the range of 70%–100% of the set point for the neutron log-rate signal whose signal value increases to the log scale. Fig. 4(d) shows the experimental results of the reactor-pool level signal that exhibits the characteristic in which the signal value falls when a problem occurs. Fig. 4(e) shows the experimental results of the pool-surface radiation signal when the signal value linearly increases. The pattern shown in Fig. 4(c)–(e) is similar to that shown in Fig. 4(a) and (b). Misjudgment and exploration occur at the beginning of the training. However, after the strategy is learned, the trend of correct judgment is maintained. The training curves shown in Fig. 4(c)–(e) indicate that the DRL can provide actions required to respond to deterioration in the neutron rate, pool level, and radiation. The training curves that converge with high reward and accuracy show the applicability of the reinforcement learning designed for safety-function monitoring task.

Fig. 5 shows the results of the DRL test when the training is completed. The DRL selects the best choice based on the Q-network from the start to the end of the test. Fig. 5(a) shows that the average reward from each test run converges to the maximum reward score. The DRL also makes correct judgment at an appropriate time, as indicated by the results of the 5,000 test episodes for the neutron log-rate signal shown in Fig. 5(b).

A design factor that affects the performance of DRL is found in this experiment, that is, the DRL exhibits different results depending on the manner of scoring the reward. In the training case with only a single reward [Fig. 6(a)], the experience of earning a good reward at the beginning exerts a great influence, and the subsequent search is less reflected in the training. On the other hand, in the training case with a variable reward [Fig. 6(b)], which adds the reward for action appropriateness and that for the response time, the DRL examines a better recommendation position that can achieve a gradual higher reward. Therefore, establishing a specific reward policy is important to achieve the desired objective when reinforcement-learning techniques are used.

5. Conclusion

This paper has introduced a support function based on the DRL technology to assist human operators in monitoring the safety functions of nuclear facilities. First, a learning framework and key elements of reinforcement learning for a target environment are established. Neural-network architecture for DRL is then presented, and a DRL algorithm is described. The experiments performed using the scenario data obtained from a safety-analysis tool demonstrate that the proposed DRL-based algorithm has the potential to help in diagnostic tasks. This work is a feasibility study for utilizing reinforcement-learning technology in nuclear systems. We expect that reinforcement learning will be better applied to operation supports such as emergency response and

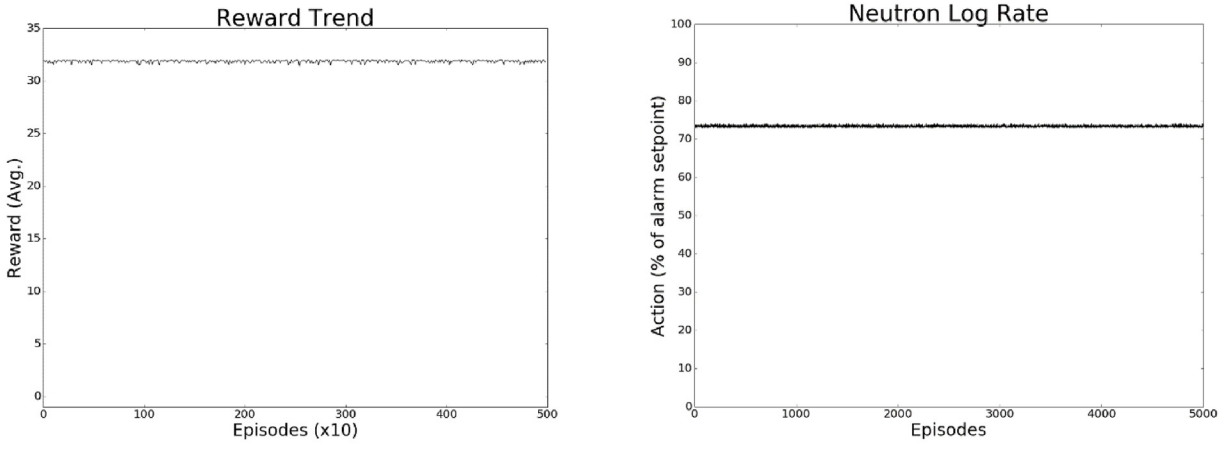
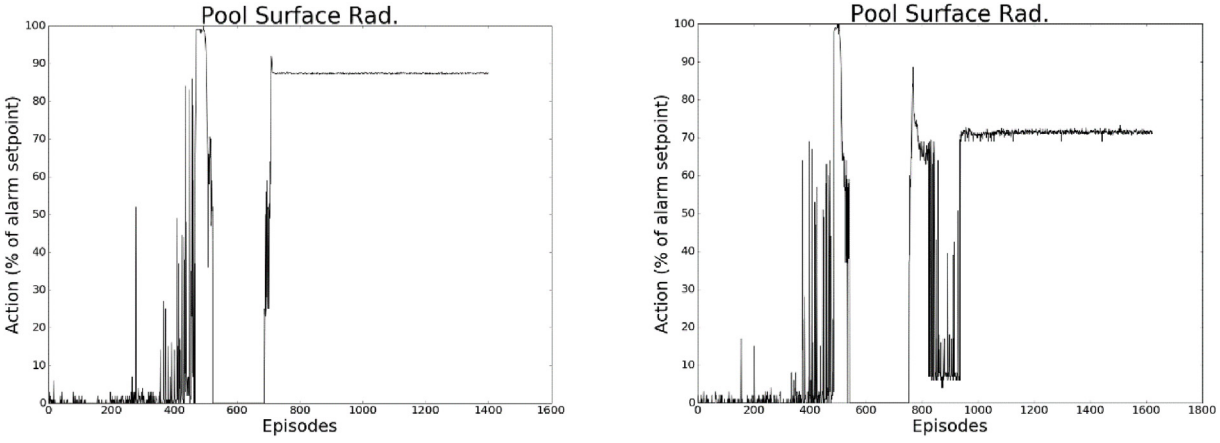


Fig. 5. Test results of the DRL algorithm.



(a) Training with a single reward (b) Training with variable reward

Fig. 6. Training results according to the reward policies.

sequence control.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2018M2B2B1065652).

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.pnucene.2019.103123.

References

Bengio, Y., 2009. Learning deep architectures for AI. Found. Trends Mach. Learn. 2, 1–127.
Buckley, J.J., Hayashi, Y., 1994. Fuzzy neural networks: a survey. Fuzzy Sets Syst. 66, 1–13.
Fantoni, P.F., Mazzola, A., 1996. A pattern recognition-artificial neural networks based model for signal validation in nuclear power plants. Ann. Nucl. Energy 23, 1069–1076.
Hinton, G.E., Salakhutdinov, R.R., 2006. Reducing the Dimensionality of Data with Neural Networks.
KAERI, 2014. Preliminary Safety Analysis Report for Kijang Research Reactor. Korea Atomic Energy Research Institute.
Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. ImageNet classification with deep convolutional neural networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems -, vol. 1. Curran Associates Inc., Lake Tahoe, Nevada, pp. 1097–1105.
Lee, S.J., Seong, P.H., 2005. A dynamic neural network based accident diagnosis advisory system for nuclear power plants. Prog. Nucl. Energy 46, 268–281.
Mitchell, T.M., 1997. Machine Learning. McGraw-Hill, Inc.
Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D., 2015. Human-level control through deep reinforcement learning. Nature 518, 529–533.
Mo, K., Lee, S.J., Seong, P.H., 2007. A dynamic neural network aggregation model for transient diagnosis in nuclear power plants. Prog. Nucl. Energy 49, 262–272.
NRC, U.S., 2001. RELAP5/MOD3.3 Code Manual Volume 1: Code Structure, System Models, and Solution Methods.
Rumelhart, E.D., McClelland, J.,L.J., 1986. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1 Foundations.
Santosh, T.V., Vinod, G., Saraf, R.K., Ghosh, A.K., Kushwaha, H.S., 2007. Application of artificial neural networks to nuclear power plant transient diagnosis. Reliab. Eng. Syst. Saf. 92, 1468–1472.
Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D., 2016. Mastering the game of Go with deep neural networks and tree search. Nature 529, 484–489.
Wei, T., Wang, Y., Zhu, Q., 2017. Deep reinforcement learning for building HVAC control. In: Proceedings of the 54th Annual Design Automation Conference 2017. ACM, Austin, TX, USA, pp. 1–6.
Yang, J., Kim, J., 2018. An accident diagnosis algorithm using long short-term memory. Nucl. Eng. Technol. 50, 582–588.
Yoo, K.H., Back, J.H., Na, M.G., Hur, S., Kim, H., 2018. Smart support system for diagnosing severe accidents in nuclear power plants. Nucl. Eng. Technol. 50, 562–569.