International Journal of Innovative Computing, Information and Control
Volume **21**, Number **2**, April **2025**

ICIC International ©2025 ISSN 1349-4198
pp. **469–480**

BAYESIAN OPTIMIZATION OF HYPER-PARAMETERS AND REWARD FUNCTION IN DEEP REINFORCEMENT LEARNING: APPLICATION TO BEHAVIOR LEARNING OF MOBILE ROBOT

Takuto Nishimura[1], Ryosuke Sota[2] and Tadashi Horiuchi[1,*]

[1]Advanced Engineering Faculty
National Institute of Technology, Matsue College
14-4 Nishi-ikuma, Matsue, Shimane 690-8518, Japan
s2311@matsue-ct.ac.jp; *Corresponding author: horiuchi@matsue-ct.ac.jp

[2]Graduate School of Science and Technology
Nara Institute of Science and Technology
8916-5 Takayama-cho, Ikoma, Nara 630-0192, Japan
sota.ryosuke.sn2@naist.ac.jp

Received August 2024; revised December 2024

Abstract. *Deep reinforcement learning is a machine learning method that combines deep learning and reinforcement learning. Deep Q-Network (DQN) is one of the typical methods of deep reinforcement learning. DQN uses Convolutional Neural Network (CNN) which can extract features from the input images. We have applied DQN method to the mobile robot navigation problem. The values of hyper-parameters, including the network structure of DQN, and the reward function used in the DQN algorithm, have been determined empirically. In this study, we attempt to optimize the values of hyper-parameters and reward function of deep reinforcement learning by using Bayesian optimization. We realized to optimize the values of hyper-parameters including the network structure of DQN, and the reward function by Optuna, a framework of Bayesian optimization. We confirmed that our method using the values of hyper-parameters and reward function obtained by Optuna have more goal achievements and fewer action steps to achieve the goal among test runs after learning than those by empirical method.*
Keywords: Deep reinforcement learning, Bayesian optimization, Hyper-parameters, Reward function, Mobile robot, Behavior acquisition

1. **Introduction.** Deep reinforcement learning offers new promise for solving complex control tasks using visual information. Developments of Deep Q-Network (DQN) [1] which is superior to human experts in several video games and AlphaGo [2] which won a human champion had strong impact all over the world. Deep reinforcement learning has remarkable features to directly learn the action policy from the high-dimensional image data by using Convolutional Neural Network (CNN) [3]. We have already realized that the mobile robot learns to acquire behaviors to reach the goal area avoiding the wall and obstacles with probability of 97.5% in the simulation environment by using DQN-based method [4]. The network structure of the DQN and the hyper-parameters were determined empirically in our previous researches [5, 6]. Here, the hyper-parameters refer to the parameters necessary to be set their values in advance, such as the learning rate, the batch size, and the kind of optimization algorithm used in neural networks. The reward function serves as the metrics for the learning process of DQN and is also determined

DOI: 10.24507/ijicic.21.02.469

empirically. The network structure of DQN, hyper-parameters, and reward function are important elements in the learning process.

In this research, we attempt to efficiently optimize the network structure of DQN, the values of hyper-parameters, and the reward function by using Bayesian optimization, for the robot navigation problem to acquire the behaviors to reach the goal area. This approach allows for the automatic and efficient optimization of crucial elements influencing learning performance in deep reinforcement learning, which were determined through trial and error in traditional researches. Furthermore, hyper-parameters of DQN and reward function often vary depending on the problem environment and problem settings. Therefore, by employing the optimization system proposed in this study, it is expected that optimized hyper-parameters of DQN and reward function for the specific problem can be obtained. As a result, in the simulation environment for the behavior acquisition problem of the mobile robot to reach the goal area avoiding walls and obstacle, we succeeded to optimize both of the hyper-parameters including the network structure of DQN, and the reward function used in the DQN algorithm, by using Bayesian optimization framework, Optuna. In addition, it was found that simultaneously optimizing the hyper-parameters of DQN and reward function yields better learning performance compared to optimizing them by two stages.

The related works are as follows. Ekundayo [7] demonstrated that, in the problem of predicting household electricity consumption, they optimized parameters of advanced CNN methods using Optuna and achieved the best performance compared to several other methods. Uemura et al. [8] developed a walking navigation system that can automatically detect stairs and steps to inform individuals with visual impairments. As part of this effort, they have utilized a Convolutional Neural Network (CNN) trained on indoor camera data to recognize stairs. The parameters of the CNN have been optimized using Optuna for maximum efficiency. Furthermore, Takaoka et al. [9] demonstrated the optimization of parameters used in advanced reinforcement learning methods with Optuna and applied action learning in a grid world environment. Like these studies, there are several works in which CNN network structures and hyper-parameters have been optimized using Bayesian optimization, but there have been few researches to realize optimizing the reward function in deep reinforcement learning.

In addition to Bayesian optimization, NAS (Neural Architecture Search) [10, 11] has also been proposed as a method to optimize the network structure of CNN. The NAS approach is also considered to be effective for optimizing the network structure and hyper-parameters of DQN, but it is considered difficult to apply it to optimizing the reward function. In contrast, Bayesian optimization is not a method specialized for optimizing network structure or hyper-parameters, but it can also be used to optimize the reward function, so in this study we use Bayesian optimization.

One method of adjusting the reward function is reward shaping, which aims to improve the learning speed in reinforcement learning by adding extra value to the normal reward value. One of the representative methods is potential-based reward shaping [12]. However, the optimization of reward functions using Bayesian optimization has not been attempted, and it has not yet been applied to robot behavior acquisition.

This paper is organized as follows. Chapter 2 explains the application of the DQN method to a robot, the Bayesian optimization techniques employed in this study, and the problem settings. In this study, we utilize the Tree-structured Parzen Estimator (TPE) algorithm, which is one of the techniques in Bayesian optimization. This chapter includes defining the problem environment, specifying the definitions of state, action, and reward spaces. Chapter 3 explains optimization targets and methods using Bayesian optimization. In this study, there are two targets: hyper-parameters of DQN and reward function

to be optimized by using Bayesian optimization framework, Optuna. Chapter 4 presents the experimental methods and results of the optimization experiments and provides a comparison of the learning performance with parameters optimized empirically. In Chapter 5, we conclude the paper and offer insights into future research directions.

2. Preliminary and Problem Statement.

2.1. DQN-based reinforcement learning.

We use DQN-based reinforcement learning method as deep reinforcement learning. DQN uses the Q-learning method which is one of the most widely-used reinforcement learning and also uses CNN in order to approximate the action-value function. The action-value function is shown by the following equation:

$$Q^{\pi}(s,a) = \boldsymbol{E}_{\boldsymbol{\pi}}\left[\sum_{k=0}^{\infty}\gamma^{k}r_{t+k}\middle| s_t = s, a_t = a\right] \tag{1}$$

where π denotes the policy $\pi = P(a|s)$, r_t is reward at time step t and γ is discount rate.

DQN uses the method called experience replay. In order to perform experience replay, the agent's experiences $e_t = (s_t, a_t, r_t, s_{t+1})$ at each time step t are stored in the data set $D = \{e_1, \ldots, e_t\}$. The data set D is also called replay memory. During learning, we apply Q-learning update rule on samples of experience (s_j, a_j, r_j, s_{j+1}) drawn uniformly at random from the data set D.

We use an improved method of DQN, which incorporates two algorithms: Double DQN and multi-step learning, into DQN in order to improve the learning performance.

2.2. Parameter optimization using Bayesian optimization.

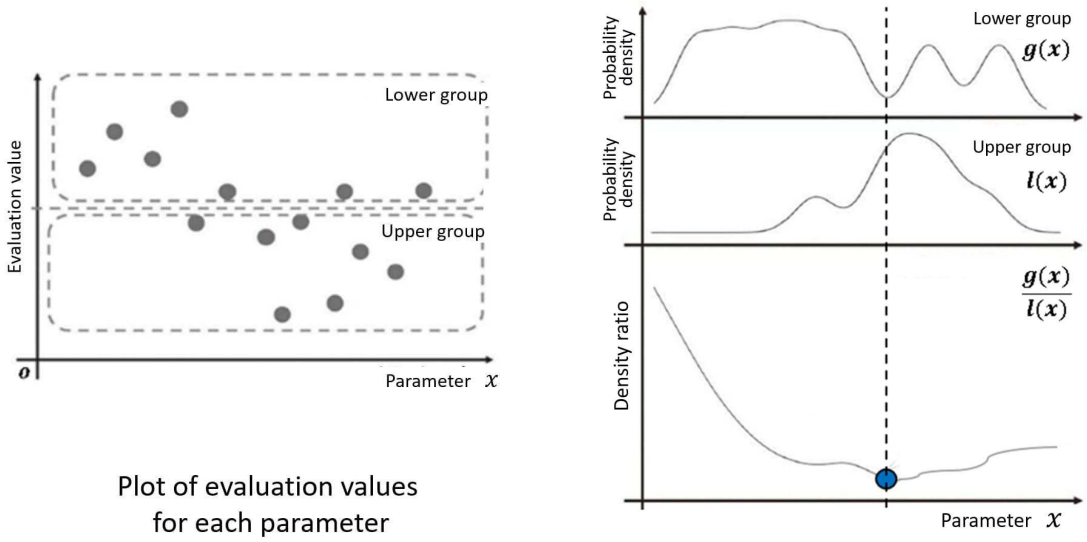
Bayesian optimization is one of the promising methods in design of experiments for optimizing experimental parameters. In this research, we apply the Bayesian optimization framework Optuna [13] in order to optimize the network structure of DQN, the hyper-parameters, and reward function.

Optuna is a method of design of experiments and it uses the TPE (Tree-structured Parzen Estimator) algorithm for Bayesian optimization. The principle of TPE is as follows. By applying Bayes' theorem to each distribution of parameter values that produced good results (the upper group) and those that produced bad results (the lower group), we can derive the parameter values which maximize the expected value of the evaluation function. Figure 1(a) illustrates the samples of evaluation values for the certain parameter and division into two groups: the upper group and the lower group in the minimization problem. We derive the probability density $l(x)$ for the upper group and $g(x)$ for the lower group respectively by kernel density estimation. By computing minimal point of the ratio $g(x)/l(x)$, we can get the next search point as shown in Figure 1(b).

The behavior learning using DQN-based method is then performed using these derived parameter values as shown in Figure 1. After the behavior learning, the parameter values are updated to maximize the expected value of the evaluation function. Then the behavior learning process is performed again using these updated parameter values. By repeating these processes, Optuna can efficiently find the optimal parameter values. On the other hand, grid search, which involves exploring all possible combinations of parameter values, is computationally expensive and time-consuming, whereas Bayesian optimization method Optuna can discover good parameter values in shorter amount of time.

2.3. Problem setting.

In this research, we use the ROBOTIS TurtleBot3 Burger robot [14] shown in Figure 2 as a mobile robot. TurtleBot3 Burger robot is a commercially available two-wheeled mobile robot which is equipped with LiDAR (Light Detection and Ranging) sensor and Raspberry Pi small computer. LiDAR is a 2D laser scanner which



(a) Division into two groups based on evaluation function value (minimization problem)

(b) Probability density

FIGURE 1. Principle of TPE in Bayesian optimization

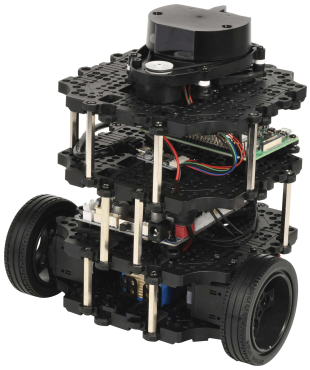


FIGURE 2. TurtleBot3 Burger

is capable of distance measurement in 360 [deg] around the device. It is easy to install ROS (Robot Operating System) software.

2.3.1. *Problem environment.* In this research, we assume the simulation environment shown in Figure 3, as the problem environment. The size of the problem environment is 1.8×1.8 [m] and it is surrounded by black walls with height of 25 [cm]. There are two goal areas and one obstacle with size of 0.4×0.4 [m] in the center.

The observation information is a set of web camera image and 12 values of the LiDAR. We attached the web camera BUFFALO BSW200MBK to the front of the TurtleBot3 Burger robot with downward at 15 [deg] angle. It captures the field of view of 120 [deg] in front of the robot. It captures color images of 1920×1080 [pixel] and we reduce the image size into 48×27 [pixel]. Whereas, 12 values of the LiDAR are obtained from 360 [deg] range measurement at 30 [deg] intervals. The LiDAR used in this research is ROBOTIS LDS-01.

Using images from the web camera and the values from the LiDAR on the robot, we aim to realize that the mobile acquires the behavior to reach the goal area while avoiding the wall and obstacle.

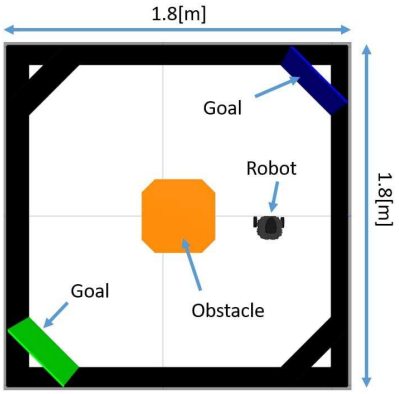


FIGURE 3. Problem environment

2.3.2. *Definition of state, action and reward.* We implemented DQN-based method using PyTorch deep learning framework and reinforcement learning library PFRL [15] by Preferred Networks. The input values of DQN are state s which consists of a camera image and 12 values of LiDAR in 12 directions. All input values are normalized between 0 and 1. The camera image is an RGB image of size 48×27 [pixel], and the LiDAR values take a maximum value of 1 when the distance is 1.8 [m].

The input state s is decomposed into the camera image and LiDAR values. By convolution process and max-pooling process for the camera image, the feature maps are obtained. Then, the values from the LiDAR are combined with the flatted feature maps. Finally, the action values $Q(s, a)$ for all candidate action a are computed through the fully-connected layer.

The robot repeatedly selects one action from three candidate actions: a_1: go straight, a_2: turn left, a_3: turn right. One action step means one cycle from the observation of state s to the execution of the selected action. By executing the action a_1 (go straight), the robot moves forward by 30 [mm]. By executing the action a_2 (turn left) or a_3 (turn right), the robot rotates by 0.28 [rad] counter-clockwise or clockwise while moving forward by 40 [mm]. 1 action step is defined as a single choice of actions.

The reward r_t is determined using the camera image and LiDAR values. The values of rewards are assigned for the following five cases:

$$\begin{cases} \text{when the robot reaches the goal area} \\ \text{when the robot collides to walls or obstacle} \\ \text{when the robot is away from walls and obstacle} \\ \text{when the robot is close to walls or obstacle} \\ \text{otherwise.} \end{cases}$$

We adopt ϵ-greedy method as action selection method. In the ϵ-greedy method, the robot randomly selects an action with a small probability ϵ; otherwise it selects the action with the highest action value.

In this research, the value of ϵ linearly decreases from 0.1 to 0 during 8,000 action steps.

In the given settings, the robot undergoes learning through 8,000 action steps. In addition, the first 1,000 action steps are for initial exploration, during which the robot does not learn and only collects the samples of experience.

During learning, when the robot collides to walls or obstacle, the robot stops at once and the robot executes the restart action. The restart function is introduced that allows the robot to restart on its own, in order to save time and effort to return the robot to the starting position.

2.3.3. *Test run.* After completing 8,000 action steps of behavior learning, we conduct a test run to evaluate the effectiveness of the learning. The test run is performed 5 times from 3 starting positions, with 5 different directions, as illustrated in Figure 4. For each test run, success is defined as reaching the goal within 500 steps. Failure occurs if there is any collision with walls or obstacles, or if 500 steps elapse without reaching the goal. The results of the five test runs are utilized to evaluate the parameters in Bayesian optimization.

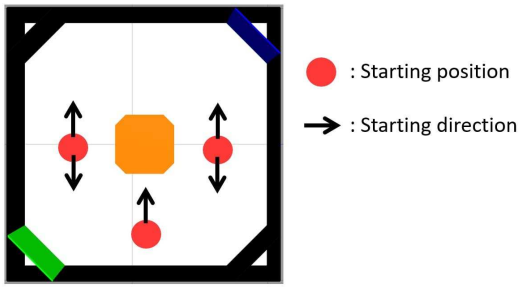


FIGURE 4. Starting positions and directions in the test runs

3. **Optimization Targets and Methods.**

3.1. **Optimization targets and search spaces.** In this study, there are two targets to be optimized using Bayesian optimization with Optuna. One is the hyper-parameters of DQN. The other is the reward function. The specific parameters and search spaces for each target will be detailed in Tables 1 and 2.

TABLE 1. Search candidate of DQN hyper-parameter optimization

Search target of hyper-parameter	Search candidate
Number of sets of convolutional and pooling layers	2, 3
Number of hidden layers in fully-connected layers	1, 2, 3
Number of hidden units	Final layer: $50 \sim 450$ Other layers: $400 \sim 800$ step size: 20
Type of optimizer	AdaGrad, AdaDelta, Adam, RMSprop
Batch size in mini-batch learning	8, 16, 32, 64, 128, 256, 512

The search candidates of DQN hyper-parameter including the network structure are shown in Table 1. For example, the number of sets of convolutional and pooling layers, the number of hidden layers in the fully-connected layers and the number of hidden units, represent the network structure of DQN. The search range of reward function optimization is shown in Table 2. We consider that the robot receives the reward in five cases as shown in this table.

3.2. **Objective function in Bayesian optimization.** In Bayesian optimization, the goodness of the target parameters is evaluated using evaluation value, and the probability density is determined. Therefore, the users need to define the method for calculating the evaluation value that serves as the optimization criterion. As DQN learns to maximize the obtained reward values, it is common to evaluate the performance based on the rewards

TABLE 2. Search range of reward function optimization

Cases where rewards are given	Search range of reward value
When robot reached the goal	$+2000 \sim 0$ step size: 100
When robot collided with wall, etc.	$-2000 \sim 0$ step size: 100
When robot is away from wall, etc.	$+20 \sim 0$ step size: 1
When robot is near wall, etc.	$-20 \sim 0$ step size: 1
Cost of each action step	$-20 \sim 0$ step size: 1

obtained during test runs. In this study, because five test runs are conducted, the average reward value obtained across these runs is used as the evaluation value.

However, in the optimization of the reward function, where the values of the reward function are modified through the search, the comparison among different reward function is not executable. Therefore, reward values are not suitable as the evaluation value. Hence, we consider a new objective function for Optuna (Bayesian optimization framework), and the output of this function is defined as the evaluation value.

We defined the objective function for Optuna to consider three factors: the number of goal achievements, the number of action steps taken to reach the goal, and the simplicity (generality) of the reward function. The objective function is represented by Equation (2). By maximizing the output value N of this objective function, we aim to find the optimal reward function through the optimization process.

$$\begin{aligned} N = {} & a \times (\text{Number of goals reached} \times 100) \\ & + b \times (500 - \text{Number of steps to reach goal}) \\ & + c \times \frac{\text{Number of reward types not adopted} \times 500}{\text{Number of reward types}(=5)} \\ \leq {} & 500 \end{aligned} \tag{2}$$

Here, "number of reward types not adopted" refers to the number of reward function with value of 0. This term allows us to search for simpler reward function. The coefficients a, b, and c determine the emphasis on the number of goal achievements, the number of action steps taken to reach the goal, and the simplicity of the reward function, respectively. These are parameters that can reflect the user's intent and have the following relationship:

$$a + b + c = 1, \quad a, b, c \geq 0 \tag{3}$$

This relationship ensures that $N \leq 500$. In this study, we defined $a = 0.5$, $b = 0.3$, and $c = 0.2$ based on the following considerations: The first term in Equation (2), the number of times the goal is reached, is the most important, followed by the second term, the number of steps to reach the goal, and thirdly, the third term, the simplicity of the reward function. Additionally, in order to properly evaluate the second term in Equation (2), the action step count is always set to the maximum value of 500 in case of collisions.

In summary, the method for calculating an evaluation value specific to each optimization target is as follows.

- In the optimization of hyper-parameters of DQN (including the network structure), the evaluation value is the average reward obtained through the test runs.

- In the optimization of reward function, the evaluation value is the output of the objective function in Equation (2) obtained through the test runs.

4. Experiments.

4.1. **Optimization experiment.** In the context of parameters optimization using Optuna for Bayesian optimization, this study focuses on two optimization targets: hyper-parameters of DQN and reward function. Taking account of the order of these optimizations, experiments are conducted for the following three methods.

Method 1: Optimize hyper-parameters of DQN first, and then optimize reward function, with 100 trials of search, respectively.

Method 2: Optimize reward function first, and then optimize hyper-parameters of DQN, with 100 trials of search, respectively.

Method 3: Simultaneously optimize both hyper-parameters of DQN and reward function, with 200 trials of search.

Here, 1 trial comprises the behavior learning for 8,000 action steps and conducting 5 test runs. Only the optimization results for method 1 are presented below, due to space limitations.

4.1.1. *Result of DQN hyper-parameters optimization.* For method 1, the optimization process begins by first optimizing the hyper-parameters of DQN among the two optimization targets. Using the optimization candidates outlined in Table 1 of Section 3.1, we employed Optuna to optimize the combinations of parameters. Conducting 100 trials of search, the following optimized results were obtained in Table 3. In addition, the optimized network structure is as shown in Figure 5. Here, the input s indicates the state (observation) in

Table 3. Result of DQN hyper-parameter optimization

Search target of hyper-parameter	Optimization result
Number of sets of convolutional and pooling layers	2
Number of hidden layers in fully-connected layers	3
Number of hidden units	1st hidden layer: 800 2nd hidden layer: 540 Final layer: 250
Type of optimizer	Adam
Batch size in mini-batch learning	8

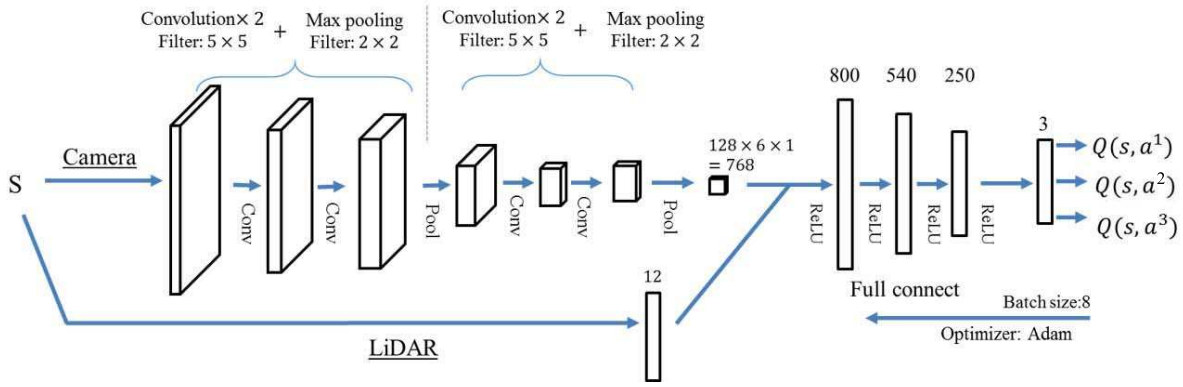


Figure 5. The network structure of DQN optimized by Optuna

reinforcement learning. In this research, the state s consists of the camera image (48×27 [pixel]) and LiDAR values (12 directions).

4.1.2. *Result of reward function optimization.* For method 1, the second stage involves optimizing the reward function. Here, the robot learns behaviors using the hyper-parameters of DQN obtained in the previous section. Using the optimization candidates outlined in Table 2 of Section 3.1, we employed Optuna to optimize the reward function. Optuna performs optimization process using Equation (2) as the objective function. Conducting 100 trials of search, the following optimized results were obtained in Table 4.

TABLE 4. Result of reward function optimization

Cases where rewards are given	Optimization result of reward value
When robot reached the goal	+1300
When robot collided with wall, etc.	0
When robot is away from wall, etc.	+19
When robot is near wall, etc.	−11
Cost of each action step	−2

The crucial point in these results is that the reward value when robot collided with wall, etc. is 0. This indicates that the simplicity (generality) of the reward function, considered in the third term of the objective function in Equation (2), is affecting the outcome.

Similar to the processes in Sections 4.1.1 and 4.1.2, optimization was performed for methods 2 and 3 using Optuna.

4.2. **Comparison of learning performance.** We compare the learning performance of four methods, namely, three methods obtained from Optuna method in the previous section for hyper-parameters of DQN and reward function, and the conventional empirically determined settings. The comparison evaluates the effectiveness of the parameters. The four methods are referred to as follows.

Method 1, 2, 3: These are explained in Section 4.1.
Method 4: Hyper-parameters of DQN and reward function determined based on empirical experience.

The approach to comparing learning performance involves conducting 10 sets of learning with 8,000 action steps and 5 test runs using the obtained parameters by each method. The results are then averaged and compared. The learning and testing procedures are consistent with those outlined in Section 2.3.

When it comes to test runs, the comparison of the average goal achievement counts through the 10 sets is illustrated in Figure 6. Additionally, the comparison of the action steps taken to reach the goal is depicted in Figure 7. From Figure 6, it is evident that the three methods utilizing Optuna achieve more goals compared to the manual approach. Additionally, it is observed that the method 3, optimized simultaneously, attains the highest number of goals. As for Figure 7, since reaching the goal earlier is preferable, lower number of action steps is considered favorable. Therefore, it can be concluded that the all Optuna-based methods reach the goal faster than the manual approach, with the method 3 optimized simultaneously, achieving the quickest goal achievement.

5. **Conclusions.** In the simulation environment for the behavior acquisition problem of the mobile robot to reach the goal area avoiding walls and obstacle, we succeeded in optimizing both of the hyper-parameters including the network structure of DQN, and the

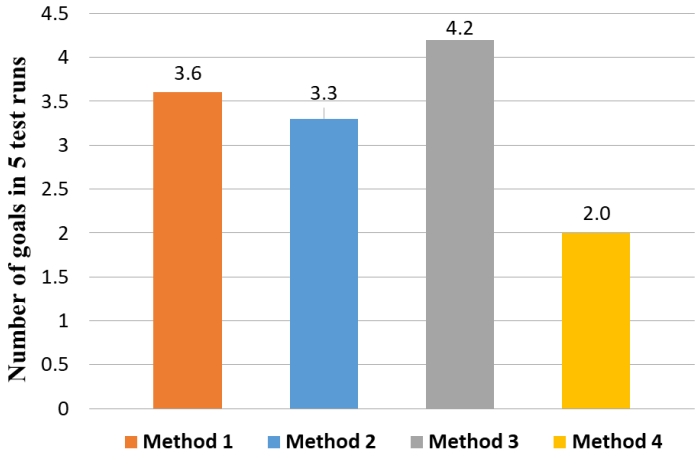


FIGURE 6. Comparison of the number of reaching goals in 5 test runs

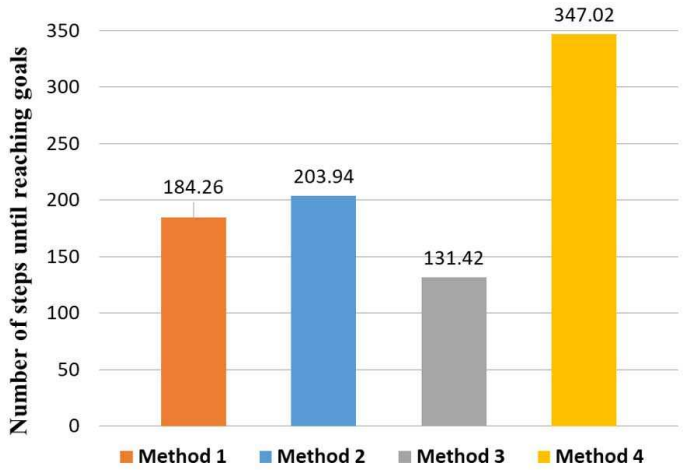


FIGURE 7. Comparison of the number of action steps until reaching goals

reward function used in the DQN algorithm, by using Bayesian optimization framework, Optuna.

We confirmed that the hyper-parameters and the reward function obtained by Optuna have higher learning ability than the empirically optimized those manually, through the simulation experiments. In addition, it was found that simultaneously optimizing the hyper-parameters of DQN and reward function yields better learning performance compared to optimizing them by two stages.

As future works, the application of the optimization system from this study's simulation environment to more practical settings, such as offices, is conceivable. The optimization system used in this study is not dependent on the environment or the format of the DQN's state inputs, making it highly versatile system. Therefore, with changes in the environment, it becomes possible to increase the DQN's state inputs and optimize the hyper-parameters of DQN and reward function accordingly. Furthermore, it is necessary to apply the hyper-parameters of the DQN and reward function obtained using Optuna from the simulation environment to the real robot environment.

REFERENCES

[1] V. Mnih et al., Human-level control through deep reinforcement learning, *Nature*, vol.518, pp.529-533, 2015.

[2] D. Silver et al., Mastering the game of Go with deep neural networks and tree search, *Nature*, vol.529, pp.484-489, 2016.
[3] Y. LeCun et al., Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, vol.86, no.11, pp.2278-2324, 1998.
[4] R. Sota, A. Fukushima and T. Horiuchi, A study on improved method for behavior acquisition of mobile robot by deep reinforcement learning, *Proc. of 2022 Annual Conference on Electronics, Information and Systems, Institute of Electrical Engineers of Japan*, pp.1505-1506, 2022 (in Japanese).
[5] R. Watanuki, T. Horiuchi and T. Aodai, Vision-based behavior acquisition by deep reinforcement learning in multi-robot environment, *ICIC Express Letters, Part B: Applications*, vol.11, no.3, pp.237-244, 2020.
[6] R. Sota, T. Nishimura and T. Horiuchi, A study on optimization of hyper-parameters in deep reinforcement learning by Bayesian optimization, *ICIC Express Letters, Part B: Applications*, vol.15, no.7, pp.723-731, 2024.
[7] I. Ekundayo, *OPTUNA Optimization Based CNN-LSTM Model for Predicting Electric Power Consumption*, Ph.D. Thesis, National College of Ireland, Dublin, 2020.
[8] A. Uemura et al., Stair recognition using CNN and step detection using depth camera, *Journal of Japan Society of Directories*, vol.21, no.1, pp.78-87, 2023 (in Japanese).
[9] S. Takaoka et al., Evaluation of hierarchical reinforcement learning methods using grid worlds, *Proc. of Game Programming Workshop 2019*, pp.172-176, 2019 (in Japanese).
[10] B. Zoph and Q. V. Le, Neural architecture search with reinforcement learning, *Proc. of the 5th International Conference on Learning Representation*, 2017.
[11] H. Pham, M. Guan, B. Zoph, Q. Le and J. Dean, Efficient neural architecture search via parameters sharing, *Proc. of the 35th International Conference on Machine Learning*, 2018.
[12] B. Badnava et al., A new potential-based reward shaping for reinforcement learning agent, *Proc. of IEEE 13th Annual Computing and Communication Workshop and Conference*, 2023.
[13] *Optuna*, Preferred Networks, Inc., https://www.preferred.jp/ja/projects/optuna/, Accessed on May 18, 2024.
[14] *ROBOTIS e-Manual*, https://emanual.robotis.com/docs/en/platform/#turtlebot3, Accessed on May 18, 2024.
[15] *PFRL*, Preferred Networks, Inc., https://github.com/pfnet/pfrl, Accessed on May 18, 2024.

Author Biography

Takuto Nishimura graduated from the Department of Control Engineering at National Institute of Technology, Matsue College and received the associate degree in 2023. He is currently a student in the Advanced Electronic and Information Systems Course of Advanced Engineering Faculty at National Institute of Technology, Matsue College. His current research interests include intelligent systems and robotics.

Ryosuke Sota graduated from the Department of Control Engineering at National Institute of Technology, Matsue College and received the associate degree in 2022. He graduated from the Advanced Electronic and Information Systems Course of Advanced Engineering Faculty at National Institute of Technology, Matsue College and received the B.E. degree in 2024. He is currently a graduate student in the Graduate School of Science and Technology at the Nara Institute of Science and Technology. His current research interests include robot learning.

Tadashi Horiuchi received the B.E., M.E. and Ph.D. degrees all from Kyoto University in 1992, 1994 and 1999, respectively. He was with Osaka University as a research associate from 1997 to 2001, and then he joined the National Institute of Technology, Matsue College, where he is currently a professor in the Department of Control Engineering and Advanced Engineering Faculty. His current research interests include intelligent systems and soft computing. He is a member of the Society of Instrument and Control Engineers, Institute of Electrical Engineers of Japan, the Japanese Society for Artificial Intelligence, the Institute of Systems, Control and Information Engineers, and the Japan Society for Fuzzy Theory and Intelligent Informatics.