Study on Stages Evaluation for 2D Action Game Generated by DC-GAN Based on Artificial Intelligence

Kotaro Nagahiro[1*], Sho Ooi[2], Mutsuo Sano[3]

[1] Graduate school of Information Science and Technology, Osaka Institute of Technology, 1-79-1 Kitayama, Hirakata-shi Osaka, 573-0171 Japan.
[2] Faculty of Information Science and Engineering, Ritsumeikan University, 1-1-1 Nojihigashi, Kusatsu-shi Shiga, 525-8577 Japan.
[3] Faculty of Information Science and Technology, Osaka Institute of Technology, 1-79-1 Kitayama, Hirakata-shi Osaka, 573-0171 Japan.

* Corresponding author. Tel.: +81 (72) 866-5301; email: m1m19a26@st.oit.ac.jp
Manuscript submitted July 14, 2020; accepted October 5, 2020.
doi: 10.17706/ijcce.2021.10.2.28-36

Abstract: Recently, research on generative adversarial networks (GANs) in deep learning has advanced rapidly. For instance, in the field of image recognition, using a GAN, the number of training data was increased. GAN have also been used to create new similar images using training images, which can help designers such as car designer, character designer, game designer and more. We have been researching a method to automatically generate a game stage based on a dot-picture using GAN. So far, we have been studying game stage generation using GAN. In this study, we consider an evaluation method for generated game stages. Specifically, we consider using the A * search algorithm to evaluate the playability of the generated game stage from the number of jumps and the clear time. As a result, the jumping count of an automatically player using the A * algorithm was 17 times for the original stage, 12 times for stage No.1, 16 times for stage No.2, and 18 times for stage No.3. Next, the clear time was 9 seconds for the original stage, 8 seconds for stage No.1, 10 seconds for stage No.2, and 11 seconds for stage No.3. In other words, we suggest stage No.1 is simpler than the original stage, and stage 2 and stage 3 are a little more difficult than the original stage.

Key words: Deep learning, generative adversarial network, 2D action game, automatic stage generation, image generation, A* search algorithm.

1. Introduction

Recently, research on generative adversarial networks (GANs) in deep learning has advanced rapidly [1]. In the field of image recognition, researchers increased the number of training images using data augmentation [2]. In other words, a GAN can generate a new image similar to a training image, thus addressing the issue of a lack of data. In addition, GAN-based methods have been applied in various ways to support a designer. For example, Reed *et al*. [3] have studied the generated method of an image from text, Yan *et al*. [4] have studied the generated of a face image from attribute words. In previous researches [5]-[10], they have studied the generation new image from the learned images.

We focus on a method to develop game stage for a game. For the generation of a game stage, in general, game designers generate some game stages during game development and players play a game on the stage made by the designers. Recently, players have developed game stages using the "Super Mario Maker" which

runs on Wii U software developed by the Nintendo Corporation. The players then play a game on the stage created by other people. However, games are developed with multiple game stages, and the development cost is expensive. Furthermore, when a user creates a game stage, the process is time consuming and restricted owing to constraints imposed by the game software. In a study of automatic generation of the game stage using GAN, Radford *et al*. [11] have been studying a game stage of "DOOM" in the first-person shooter (FPS) game. In this study, they have conducted an automatic generation of the 3D game stage using the image of the viewpoint seen from directly above.

We have been studying the generation of game stages for 2D action scroll games as an application of GAN [12], [13], and describe an evaluation using A* search algorithm to game stages generated by GAN in this study as shown in Fig. 1. In this study, we consider an evaluation method for generated game stages. Specifically, we consider evaluating the playability of the generated game stage like a game using the A* search algorithm. We evaluate the stage using the A* search algorithm in the same way as Vanessa Volz *et al*. [14].

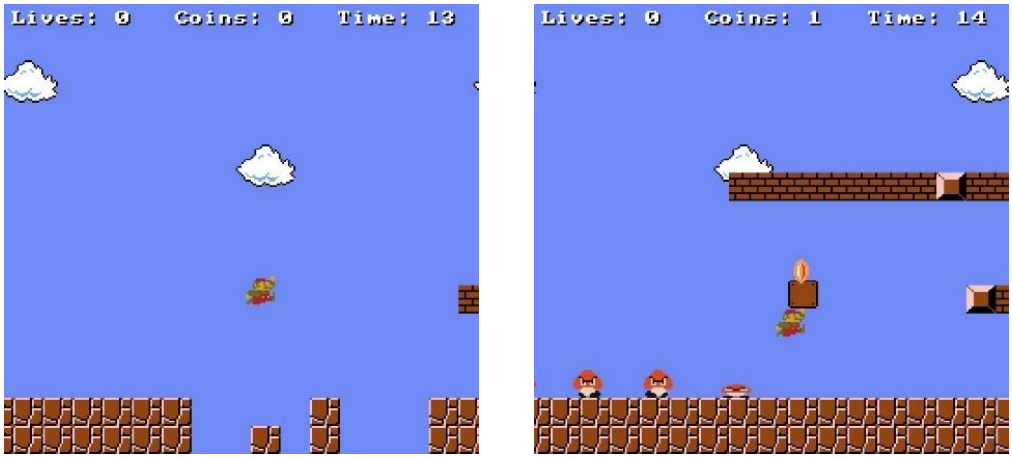


Fig. 1. An evaluation using A* search algorithm to game stages generated by GAN.

2. Generating Method of a Game Stage

In this section, we explain a generation method of game stages using GAN. Fig. 2 shows a flow to generate game stages.

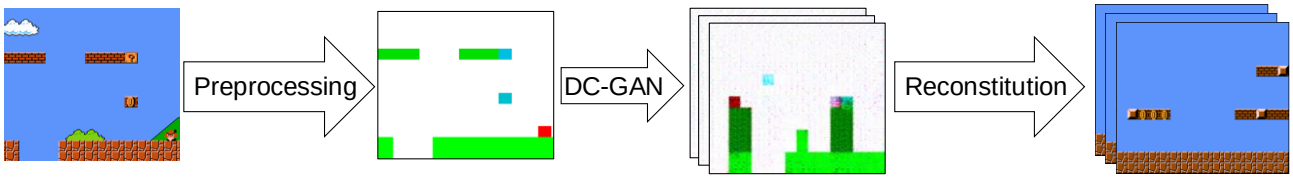


Fig. 2. Increasing learning images.

2.1. About DC-GAN

There are several types of GAN using a convolution neural network (CNN), such as the Laplacian pyramid of adversarial networks (LAP-GAN). The LAP-GAN was proposed by Denton *et al*. [15]; it can create a high-resolution image. However, the LAP-GAN needs multiple networks and processing time. Therefore, we use the DC-GAN proposed by Radford *et al*. [16], which can generate high-resolution images without using multiple networks. DC-GAN has four features. One is that it does not use a pooling layer; the second is that it does not use a fully connected layer, the third is that it uses batch normalization, and finally, it uses the leaky rectified linear unit (Leaky ReLU).

2.2. Method for Increasing Learning Image

GANs require many learning images for training; however, each game has a limited number of game stages. Hence, we propose a method of dividing a long game stage by constant sliding, as shown in Fig. 3, thus increasing the number of learning images.

A 2D action game has a long vertical or horizontal stage. Therefore, we slide vertically or horizontally to divide the game stage. In this study, the window size is 256 px × 224 px, and the sliding window size is 16 px, which is one slide of a dot picture. Fig. 3 shows an example of a sliding windows using "Super Mario Brothers."

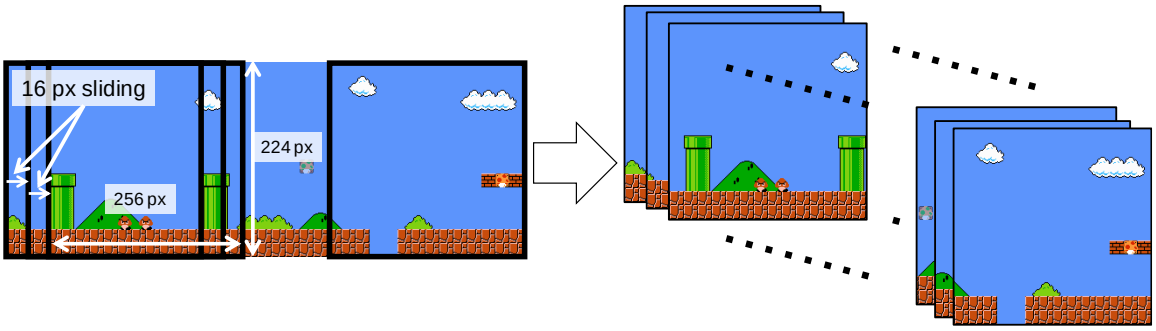


Fig. 3. Increasing learning images.

2.3. Issues Experienced with GAN Using Dot-Picture

For learning dot pictures, we target nine game stages (1-1，2-1，4-1，5-1，5-2，7-1，8-1，8-2 and 8-3), all of which have stages with a ground and blue background, from the game "Super Mario Brothers", and increased the number of learning images to 2,080 using the previously described method. In other words, we do not target underground game stages, sea stages, and sky stages. We then performed learning using the DC-GAN. The result is as shown in Fig. 4. Learning is performed for a total of 100 epochs.

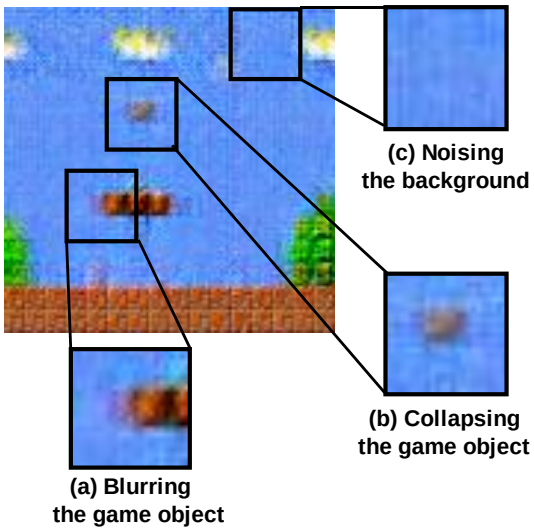


Fig. 4. Problems of game stage using DC-GAN.

For learning using the dot-picture of original game stages, as shown in Fig. 1, some problems such as collapsing of the game object and noise in the background are observed. The issues shown in Fig. 4 are described below:

(a) Blurring the game object

The generated game object appears translucent. In this case, the motion of the game object becomes

unclear.

(b) Collapsing the game object

The game object of the dot-picture is a quadrangle. However, the game object is generated incorrectly by the DC-GAN and causes gameplay issues.

(c) Noising the background

The background of this game is blue with dots of different colors in some places. These noises may affect the game.

To solve the above problems, in this research, we focus on training images before learning by preprocessing.

2.4. Proposed the Method of Preprocessing

To solve the aforementioned problems, we replace some objects in the original images with pure colors (RGB) and delete unnecessary color information. The purpose of this pre-processing is to simplify objects and backgrounds and to reduce the image blur and noise generated by the DC-GAN. Furthermore, we can easily modify the deformation of the object. The specific process consists of three steps;

(a) Classification of game objects

We classify the objects in the game into three types of "enemy," "block", and "item" and map them to “R,” “G”, and “B” in terms of color information in RGB, respectively.

(b) Quantization

We subdivide the behavior of each of the three types of objects and assign numbers between 0 and 255, equally spaced.

(c) Translation

The value assigned to the game objects is defined as a value on the RGB color scale. However, game objects not related to the player character are converted to white. For instance, the background is replaced white.

Table 1 shows the conversion results of preprocessing in "Super Mario Bros." This conversion table excludes objects that do not use for learning and cannot apply in the Mario AI Framework. Fig. 5 shows stage 1-1 in "Super Mario Brothers" after preprocessing using the conversion table.

Table 1. Type of Object in “Super Mario Bros”

Enemy (R)			Block (G)			Item (B)		
dot	color	pixel	dot	color	pixel	dot	color	pixel
		255			255			255
		213			204			170
		170			153			85
		128			102			
		85			51			
		43						

2.5. Restoring to a Playable Game Stage

We restored the game object so that it can be played against the quantized game object shown in Fig. 6 [7]. Specifically, the pixel values within the range of the generated object are examined and restored to those close to the original object based on Table 1. However, if the pixel values in the range are averaged, it becomes the median value of all the pixels, and an object with a pixel value that does not appear in the range may be selected. Therefore, we calculated which object is closest to each pixel in the range, and decided the object to be restored by their majority vote. As a specific process, the RGB value for each 1px is

set as a three-dimensional vector, and the vector is closest to the RGB value based on Table 1 by calculating the Euclidean distance. This calculation is performed for all pixels. Fig. 5 shows the restored a game object.

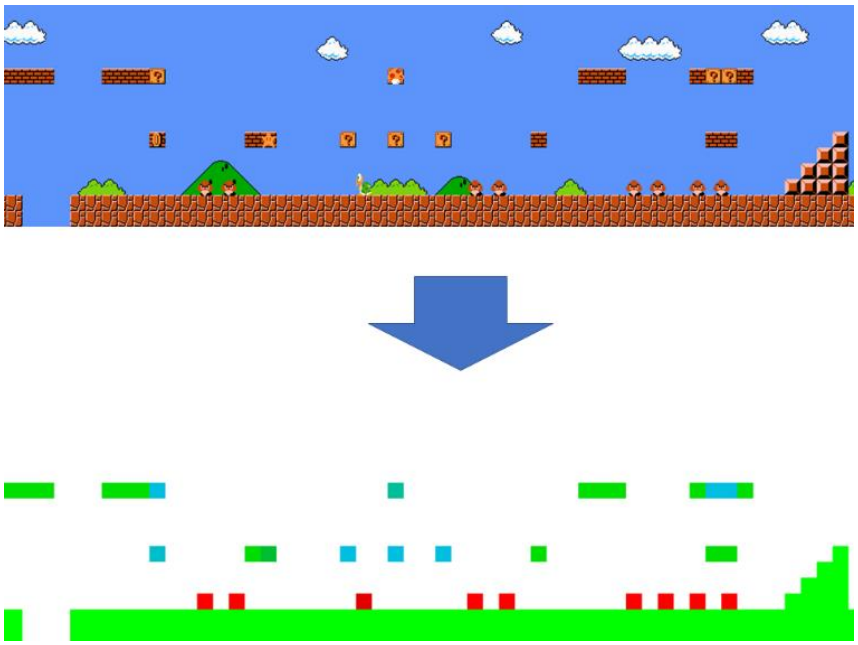


Fig. 5. The result of preprocessing.

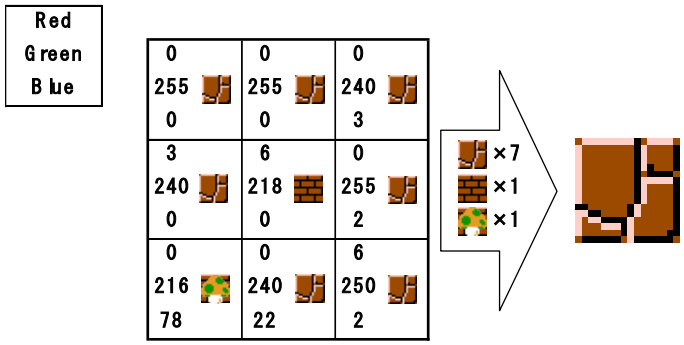


Fig. 6. The result of the restored a game object.

2.6. Playing Evaluation of Game Stages

In order to evaluate how similar the restored game stage is to the original stage, we evaluate the number of jump actions and the time until the stage is cleared, as in the study by Vanessa Volz *et al*. [8]. Specifically, AI using A* search algorithm plays and evaluates the generated game stages using DC-GAN. The A* search algorithm is a widely used best-first graph search algorithm that finds a path with the lowest cost between a pre-defined start node and one out of possibly several goal-nodes [17]. This algorithm is used to search for the best path in game state space, which is different from simply searching in the space of Mario's positions and requires that a fairly complete simulation of the game's dynamics is available to the search algorithm [18].

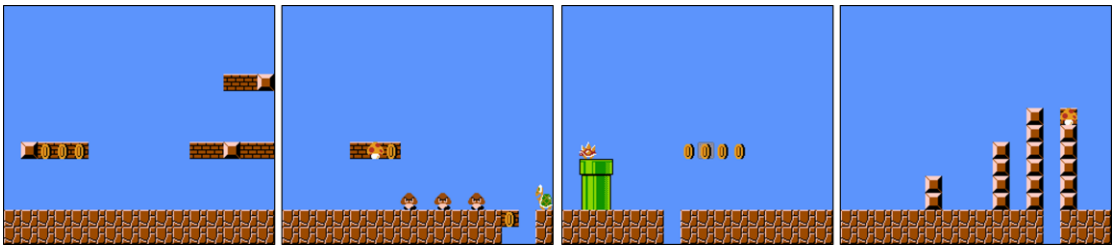
3. Experiments

In this study, we chose "Super Mario Brothers" for horizontal scrolling, because the numbers of enemy characters and items are few in this game, and stage data of some stages is large. In this experiment, we selected 13 stages of the ground type (1-1, 2-1, 3-1, 3-2, 4-1, 5-1, 5-2, 6-1, 6-2, 7-1, 8-1, 8-2 and 8-3). To

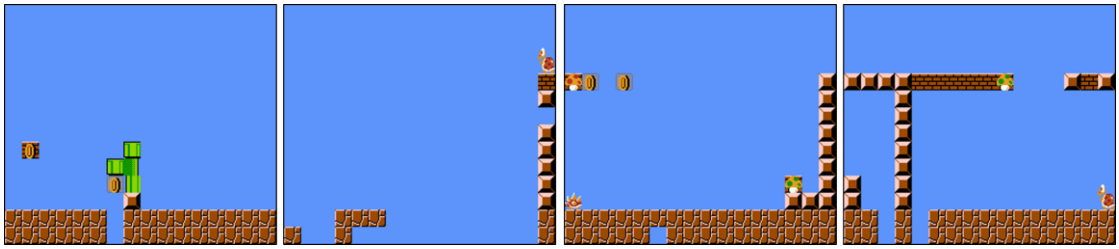
increase the number of images, the sliding window was set horizontal, and the system generated 2,912 training images. After that, we generated 6,336 stages by the method shown in Chapter 2 as shown in Fig. 6. For this generated stage, we examined the possibility of game play using the A* search algorithm. In this study, the evaluation with A* search algorithm compares the original stage 1-1 and the three-game stages generated by DC-GAN.

4. Results and Discussion

Fig. 7 shows game stages generated by GAN. The generated game stage had an object's layout that appears in the "Super Mario Bros." stage, and GAN could generate the game stage with the characteristics of the original game stage. However, we thought that some of the generated game stages could not clear, such as big holes, high walls, and dead ends, as shown in Fig. 7 (b), and excluded those game stages from the evaluation. Fig. 8 shows three kinds of game stages generated by GAN, Fig. 9 shows the jumping count and the clear time using A* search algorithm on the original stage 1-1 and three kinds of game stages. We referred to the study of Vanessa Volz *et al.* [8]. Fig. 1 shows the playing scene using A* search algorithm.



(a) A good stage (subjective evaluation).

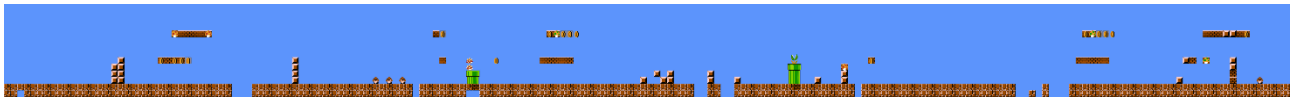


(b) A bad stage (subjective evaluation).

Fig. 7. Parts of game stage the generated by GAN.

(a) Game stage No.1.

(b) Game stage No.2.



(c) Game stage No.3.

Fig. 8. The results of the game stages generated by GAN.

As a result, the jumping count of an automatically player using the A * algorithm was 17 times for the original stage, 12 times for stage No.1, 16 times for stage No. 2, and 18 times for stage No.3. Next, the clear

time was 9 seconds for the original stage, 8 seconds for stage No. 1, 10 seconds for stage No. 2, and 11 seconds for stage No. 3. In other words, stage No. 1 is simpler than the original stage, and stage 2 and stage 3 are a little more difficult than the original stage, however, there is little difference in a game level.

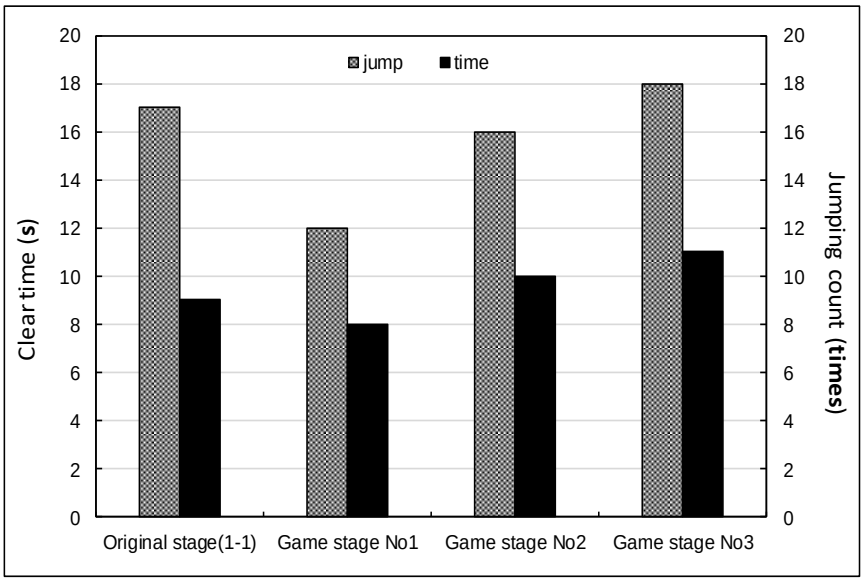


Fig. 9. Evaluation Results of game stages using A* search algorithm.

As a reason, we think that there was no significant difference in the difficulty level because we reduced the number of enemies and objects in the conversion rule shown in Table 1. In the future, we think that the difficulty level can be controlled by adding untargeted objects and untargeted enemies.

5. Conclusion

In this study, we evaluated the difficulty level of the game stage generated by GAN using the A * search algorithm. Specifically, we compared the jumping count and the clear time for the three types of game stages and original stage 1-1 and GAN. As a result, we confirmed that the difficulty level of the game stage generated this time is almost the same as the difficulty level of the original game stage 1-1.

In the future, we plan that conduct a questionnaire with people playing the generated game stage, evaluate a subjective of the game, such as a fun, a difficulty by having people play the generated game stage.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Kotaro Nagahiro conducted the research and wrote this paper; Sho Ooi and Mutsuo Sano provided guidance.

References

[1] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial networks. *Advances in Neural Information Processing Systems*, 2672-2680.

[2] Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., & Le, Q. V. (2018). AutoAugment: Learning augmentation policies from data. arXiv:1805.09501.

[3] Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., & Lee, H. (2016). Generative adversarial text to image synthesis. *Proceedings of the International Conference on Learning Representations*.

[4] Yan, X., Yang, J., Sohn, K., & Lee, H. (2016). Attribute2image: Conditional image generation from visual attributes. *Proceedings of the European Conference on Computer Vision* (pp. 776-791).

[5] Zhang, R., Isola, P., & Efros, A. A. (2016). Colorful image colorization. *Proceedings of the European Conference on Computer Visio*n (pp. 649-666).

[6] Iizuka, S., Simo-Serra, E., & Ishikawa, H. (2016). Let there be color!: Joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics, 35(4)*.

[7] Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired imagetoimage translation using cycle-consistent adversarial networks. arXiv preprint arXiv:1703.10593.

[8] Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2016). Image-to-image translation with conditional adversarial networks. arXiv preprint arXiv:1611.07004.

[9] Kim, T., Cha, M., Kim, H., Lee, J., & Kim, J. (2017). Learning to discover cross-domain relations with generative adversarial networks. arXiv preprint arXiv:1703.05192.

[10] Antipov, G., Baccouche, M., & Dugelay, J. L. (2017). Face aging with conditional generative adversarial networks. arXiv preprint arXiv:1702.01983.

[11] Giacomello, E., Lanzi, P. L., & Loiacono, D. (2018). DOOM level generation using generative adversarial networks. arXiv:1804.09154.

[12] Nagahiro, K., Ooi, S., & Sano, M. (2019). Study on stage generation for 2D action game based on DC-GAN. *Proceedings of the 7th IIAE International Conference on Intelligent Systems and Image Processing*.

[13] Nagahiro, K., Ooi, S., & Sano, M. (2019). Reconstitution of Simple game stage generated by DC-GAN for game playing. *Proceedings of the Media Computing Conference Engineers of Japan*.

[14] Volz, V., Schrum, J., Liu, J., Lucas, S. M., Smith, A., & Risi, S. (2018). Evolving Mario levels in the latent space of a deep convolutional generative adversarial network. *Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 221-228).

[15] Denton, E., Chintala, S., Szlam, A., & Fergus, R. (2015). Deep generative image models using a laplacian pyramid of adversarial networks. *Advances in Neural Information Processing Systems*, 1486-1494.

[16] Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv:1511.06434.

[17] Hart, P., Nilsson, N., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics, 4(2)*, 100–107.

[18] Karakovskiy, S., & Togelius, J. (2012). The Mario AI benchmark and competitions. *IEEE Transactions on Computational Intelligence and AI in Games (TCIAG), 4(1)*, 55-67.

Kotaro Nagahiro received his B.S. degree in information science from Osaka Institute of Technology in 2019. Currently, he is a master student with Graduate School of Information Science and Engineering, Osaka Institute of Technology, Japan. His research interests include image closing, deep learning, generative adversarial network, and game studies. He is a member of Digital Games Research Association of Japan (DiGRA Japan).

Sho Ooi received his B.S., M.S., and Ph.D. degree in information science from Osaka Institute of Technology in 2013, 2015, and 2018, respectively. Currently, he is an assistant professor with Faculty of Information Science and Engineering in Ritsumeikan University, Japan. His research interests include computer vision, cognitive science, pattern recognition, and education technology. He is a member of Information Processing Society of Japan (IPSJ), the Institute of Electronics, Information and Communication Engineers (IEICE), The Institute of Image Electronics Engineers of Japan (IIEEJ), Robot Society of Japan (RSJ), and IEEE.

Mutsuo Sano received his B.S., M.S., and Ph.D. degree in engineering from Kyoto Institute of Technology in 1981, and Kyoto University in 1983, and 1995, respectively. Between April, 1983 and March, 1985, He was engaged in the research and robot vision, image processing and contents management of Nippon Telegraph and Telephone Corporation (NTT). Currently, he is a professor and a dean with Faculty of Information Science and Technology in Osaka Institute of Technology, Japan. His research interests include computer vision, cognitive science, cooking media, cooking support, and human robot communication. He is a member of Information Processing Society of Japan (IPSJ), the Institute of Electronics, Information and Communication Engineers (IEICE), The Institute of Image Electronics Engineers of Japan (IIEEJ), Robot Society of Japan (RSJ), The Japanese Society for Artificial Intelligence (JSAI), The Institute of Image Information and Television Engineers (ITE), and IEEE.