Hindawi
Mobile Information Systems
Volume 2022, Article ID 9984617, 8 pages
https://doi.org/10.1155/2022/9984617

Research Article

A Study on the Agent in Fighting Games Based on Deep Reinforcement Learning

Hai Liang and Jiaqi Li

Faculty of Humanities and Arts, Macau University of Science and Technology, Taipa, Macau, China

Correspondence should be addressed to Jiaqi Li; jiqli@must.edu.mo

Received 14 June 2022; Revised 4 July 2022; Accepted 7 July 2022; Published 31 July 2022

Academic Editor: Praveen Kumar Reddy Maddikunta

In this study, an end-to-end noninvasive frame system available for varieties of complete information games was first implemented. After altering some codes, the system can be rapidly and effectively applied in a series of complete information games (e.g., NI-OH, The King of Fighters, and Maple Story), other than the fighting games. The fighting game Street Fighter V was selected as the experimental subject to explore the behavioral strategies of the agent in fighting games and verify both the intelligence and validity of deep reinforcement learning in the games. During the experiment, a double deep-Q network (DDQN) was adopted to train the agent in the form of fighting against in-game AI. In this process, 2590 rounds of agent training were conducted, generating a winning rate of nearly 95%. Then, the trained model underwent a series of tests, achieving winning rates of 100% (10 rounds), 90% (30 rounds), and 96% (50 rounds).

1. Introduction

When deep learning and reinforcement learning are combined and used in games, it gradually turns out that they outperform a vast majority of human players in turn-based games such as chess and card in a fixed game environment. However, no such outstanding performance is achieved in fighting games for a reason that such games are featured with a short decision-making time, a broad decision space, and greatly varied strategies. To overcome these defects, researchers need to profoundly investigate a certain fighting game and then adjust the design according to specific characteristics of different games.

In Street Fighter V, there are eight essential elements of interface information: health points, power bars, rage points, and vertigo points of both sides, as shown in Figure 1. The action space includes 12 actions falling into the following three types of action spaces: (1) a move action space of jump, squat, left movement, and right movement; (2) a hand attack action space of long-range hand attack, a mid-range hand attack, a short-range hand attack, and a combined hand attack; and (3) a leg attack action space of long-range leg attack, mid-range leg attack, short-range leg attack, and combined leg attack.

For a player of fighting games, one or part of the principles below should be abided by in combat:

For an agent, it may either follow the strategies of traditional players or lay down its path to realizing the purpose of enabling players to avoid or learn from its playing methods. We hold a high expectation for this.

The main contributions of this study are as follows.

(1) Developed an end-to-end noninvasive frame system available for multiple complete information. By virtue of this system, researchers only need to alter or add codes of relevant information (e.g., health points and rage points of characters), based on the characteristics of the game. The specifics of this system (for academic research only and not for commercial use) will not be iterated here, and you may refer to another article of the authors if needed.

(2) Succeed in enabling the agent firing varieties of skill sets by merely performing hard coding of its move and attach actions, rather than its skillsets.

Figure 1: A scene from Street Fighter V.

(3) Verified the intelligence of DDQN, a deep reinforcement learning algorithm, in fighting.

Moreover, related works, algorithms, their improvements, and relevant experiments, as well as conclusions and the future work of this paper, will be also introduced.

2. Related Works

Yoon and Kim successfully applied DQN in visual fighting game AI competitions. As the number of actions is reduced to 11, the corresponding experimental results also show that DQN as a deep reinforcement learning algorithm still has great potential in real-time fighting games [1].

Pinto and Coutinho designed a scheme of combining hierarchical reinforcement learning (HRL) with Monte Carlo tree search (MCTS) to train the agent in fighting game AIs (FTGAIs). At last, the learning strategy produced by the agent is as good as that of the champion in terms of success rates [2].

Based on the FightingCE platform, DQN [3] was successfully applied by Y. Takano et al. It was proved that the strategy of double-agents is obviously superior to single-agent with a success rate of 30% higher [4]. In addition, they also applied curiosity-driven intrinsic reward into reinforcement learning of fighting games. As demonstrated in experimental results, the learning capability of the proposed AI was above that of the actor-critic model [3]. However, it was believed, according to experience by Inoue et al., that AI cannot always effectively win against a rival that it has never met before. In other words, AI should be constantly trained in competitions [5].

2.5 D fighting games not only show fuzziness in the visual appearance of characters, such as depth or height but also require a specific sequence of continuous actions. For this reason, it becomes extremely difficult for network design. Based on asynchronous advantage actor-critic (A3C) network and *Litter Fighter 2* (LF2), Li et al. designed a novel network A3C+. Therefore, the skill sets of a fight can be observed through the game-related information features and a recursive layer, achieving a winning rate of 99.3% [6].

In a study by Ishii et al., an independently developed MCTS variant puppet-master was selected based on the FightingICE platform to control all characters in a game. Successfully, all characters were capable of moving according to the given character's features [7]. The study has great significance in proving whether the proposed method is intelligent enough to entertain the public [7].

Through Fighters Arena, Bezerra et al. implemented an agent that utilizes a fusion architecture of learning, cognition, and navigation (FALCON) and ARAM, achieving a winning rate of 90% in-game AI in a fixed action pattern [8].

Depending on the FightingICE platform, Kim et al. utilized self-play and MCTS to create an AI agent. Their research analyzed configurations of reinforcement learning, such as reward set-up and rival compositions, and proposed new performance indexes. In the course of experiments, AI constructed outperformed other AI and obtained a winning rate of 94.4% [9].

Oh et al. selected the game Blade & Soul as the experimental environment and designed two data skipping techniques: inaction and keeping moving. After the successful training in 1V1 fights, the AI agent was arranged to compete with five professional players. At last, the agent achieved a winning rate of 62% and generated an AI agent possessing three fighting styles [10].

Regarding fighting games, each laboratory team has its unique algorithm design and improvements; and most of them have made abundant accomplishments.

3. Methods

3.1. Introduction to DDQN. In DDQN, a refers to action, γ to attenuation factor, $Q\prime$ to policy-based network, Q to the target network, $\theta\prime$ to a parameter of $Q\prime$, θ to a parameter of Q, and ϕ to the state r for rewords.

In a traditional DQN, the target value q can be calculated by the following equation:

$$y_j = r_j + \gamma \max_{a\prime} Q'\left(\phi_{j+1}, a'; \theta'\right). \tag{1}$$

Considering that this equation may lead to over-estimation and thus a rather large deviation, DDQN was thus adopted here. Based on this network, an action corresponding to the maximum value of q was first figured out in a policy-based network. Subsequently, q values of the selected actions were calculated in the target network. Eventually, the following equation is obtained:

$$y_j = r_j + \gamma Q'\left(\phi_j, \arg\max_a Q\left(\phi_j, a, \theta\right); \theta'\right). \tag{2}$$

From (2), we can clearly see that the following parameters are extremely critical: action a, reward r corresponding to action, and attenuation factor y.

(1) The design of action is related to the way in which the intelligent experience feeds back the current environment and then affects the follow-up process. For example, giving an agent only one action and N actions is a completely different effect; for example, only giving agents mobile or only giving agents attack, these are two completely different effects. Therefore, this is the key parameter.

(2) The reward r corresponding to the action will affect the choice of agents to a certain extent. From the algorithm, we find that this is a constant term. Too large constant term will make the agent prefer to choose this action. Too small constant term will make the agent give up this action. Even if there is model free, it also deeply affects the effectiveness of the algorithm. Therefore, after many experiments, we finally determined the final reward setting in 2.1.

(3) As for the attenuation factor y, it is a parameter related to the future. A lower value means that we pay more attention to the current reward. A higher value means that we pay more attention to future rewards. Generally, the value will not be set to 0 or 1 but will be set to about 0.9. In terms of coding, the normalized data and the state of the next moment were simultaneously inputted into the policy-based and the target networks. Action a with the highest value in the policy-based network was labeled, then value_q corresponding to this action a was located in the target network. By calculating the reward of value_q, value_Q was obtained. The state of the previous moment was inputted into the policy-based network, and the value of the corresponding action a was altered by using value_Q. In this way, the sample was trained.

4. Reward Set-Up

4.1. Current-State Reward for Characters Controlled on Our Side. (1) In the context where the next moment's health point of a character is greater than or equal to its health point of the current moment, a reward of the difference between such two health points multiplied by 1.0 is endowed. (In fact, the best circumstance is that the health point reward of the characters on our side is endowed with 0.) (2) Vertigo point of the next moment is above or equal to that at the current moment, the reward endowed is the difference of vertigo points at both moments multiplied by (−0.1). (This manifests that the characters on our side are being attacked by in-game AI.) (3) If the rage point of a character at the next moment is greater than or equal to that at the current moment, the reward should be their difference multiplied by 0.1, signifying that characters on our side are attacking or being attacked by NPCs from the adverse party. (4) When the power bar of the next moment becomes higher than or equal to that of the current moment, the reward endowed to the corresponding character should be their difference multiplied by 0.1. (This indicates that the characters on our side are attacking or being attacked by NPCs from the adverse party.) (5) When the final health point of characters is 0 and as-failed conditions on our side are satisfied, an extra reward for failure is endowed that is −500.

4.2. Current-State Reward for In-Game AI. (1) If the next moment health point of an NPC from the adverse party becomes lower than equal to that of the current moment, a reward of the difference between such two health points multiplied by (−0.4) is endowed. (This indicates that characters on our side are attacking the NPCs.) (2) When the next-moment vertigo point of NPC from the adverse party is above or equal to that at the current moment, the reward endowed is the difference of vertigo points at both moments multiplied by 0.1. (This indicates that the NPCs are being attacked.). (3) If the rage point of a character at the next moment is greater than or equal to that at the current moment, the reward should be their difference multiplied by (−0.1), signifying that NPCs from the adverse party are attacking or being attacked by characters on our part. (4) When the next-moment power bar becomes higher than or equal to that of the current moment, the reward endowed should be their difference multiplied by (−0.1). (This indicates that NPCs from the adverse party are attacking or being attacked by the characters on our side.). (5) When the final health point of NPCs from the adverse party is 0 and victory conditions on our part are satisfied, an extra reward for winning is endowed, and that is 500.

Through summation of the current-state reward of characters controlled on our side and those of the in-game AI, the current-moment reward for the agent can be obtained.

Besides, another two reward conditions were also established. First, the health points of characters in two consecutive frames are compared; if they are unequal, the agent should be rewarded with −50. Second, if the health points in two consecutive frames are consistent, a reward of +20 will be given. During the experiment, the agent became inclined to jump, rather than attack in training rounds 150 to 450. Considering this, we had to delete the second reward condition and retain only the first reward condition.

5. A Traditional Network Architecture

Classic games, in which deep learning is applied for experiments were analyzed, including Mario, Flying Chicken, and Space Invaders. These games are easy to operate and all outputted in a single form (e.g., up, down, left, right, left + attack, or right + attack). And the output results are embodied in N columns and 1-row, as presented in Figure 2.

During agent design and implementation at the very beginning, the above thoughts were used for reference, and the traditional network architecture was adopted. To be concrete, convolutional neural network (CNN) was selected for depth calculation, and output in a single form was used. In other words, the selection was done among all behavior actions. However, the agent in this scenario may repeat a single action that can produce the maximum benefit (jump repeatedly or make repeated long-fist/long-leg attacks), and no defense or movement would be performed. Facing in-game AI, the winning rate of such an agent is dramatically low.

Through analysis, defects of such network output can be described as follows. The final agent may take an optimistic view on actions due to the existence of TD errors. In other words, it becomes increasingly inclined to do the actions of the maximum values, eventually leading to the repetition of a certain action. Taking Space Invaders, for example, the

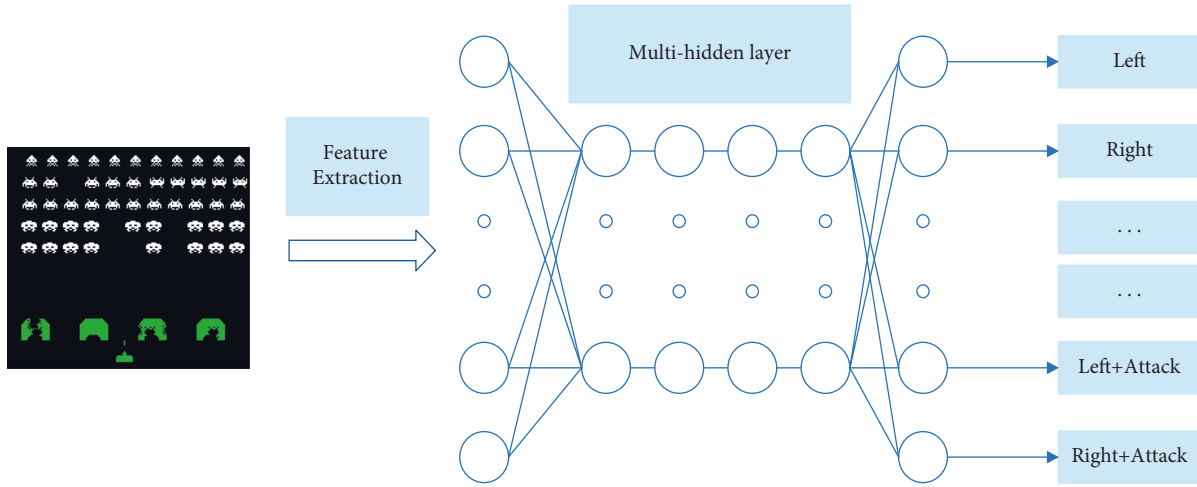

Figure 2: Network architecture of agent in Space Invaders.

combined action of attack and left or right movement is infinitely repeated, which is the same case for the agent provided with this architecture.

6. Network Architecture Improvements

In ordinary games, RTS games require both hands to control the keyboard and mouse at the same time. The keyboard corresponds to moving, jumping, lying down, and other necessary menu operations. The mouse usually corresponds to a simple and single attack (shooting) and aiming. Unlike such games, fighting games require players to operate the keyboard with both hands at the same time, corresponding to movement, jumping, squatting, and attack, respectively. However, the game mode at this time is no longer a single form of attack but contains multiple forms of attacks (basic attacks, such as fists and feet, combined skill attacks). This is the particularity of this kind of game we mentioned before. There are many decision-making needs and rich strategic changes. At the same time, this kind of game requires players to decide the output of multiple actions in a very short time. That is to say, at a certain moment, the action that needs to be made is a composite action. This is not a simple discrete form. A composite action contains multiple game character actions. In specific cases, it is often accompanied by jumping or attack during movement, or both, rather than simply completing in movement.

Therefore, output forms of the deep neural network were altered in this study. In detail, the action spaces were regrouped, and every five actions were considered as one group. The new groups include a left/right movement group consisting of left, right, left, right, and null; a jump/squat group covering jump, squat, jump, squat, and null; a hand attack group of short-range hand attack, mid-range hand attack, long-distance hand attack, composite hand attack, and null; and a leg attack group of short-range leg attack, mid-range leg attack, long-range leg attack, composite leg attack, and null. Thus, the original 12-column 1-row output was changed into that of 4 columns and 5 rows, as given in Figure 3.

For each frame, an action combination was selected, that is to select behavior actions corresponding to each of the left and right movement space, jump/squat space, hand attack space, and leg attack space. The selected actions were then combined and executed using thread. When a single action combination is implemented, a programmed time delay needs to be conducted for each action before the next frame is inputted into the network.

Finally, agent evolvement was accomplished and the agent was capable of moving while attacking or defending or selecting no actions.

7. Improvements of Other Experimental Procedures

During the experiment, it is found that once the memory pool update reaches a certain node, the policy-based network training update in this scenario may enable the agent to be in a dead state, performing no movement or attacking actions; but, NPCs from the adverse party are in the status of moving and attacking. This can lead to data missing. Given this, memory pool update was no longer considered a condition of policy-based network training update. In this case, a training update for the policy-based network was conducted after each round of fights. In terms of the target network update, it was performed every 4 rounds.

8. Experiment

In addition to the influence of the algorithm, in order to get better performance, the more feasible AI operation method. In the process of training, through many attempts and combined with the characteristics of the game, we give the following parameters, as shown in Table 1.

(i) Replay memory size indicates the size of the experience pool. It needs to be designed according to the number of actions of the game and the structure of the network. If it is like the traditional network structure model, 10000 experience pool capacity is

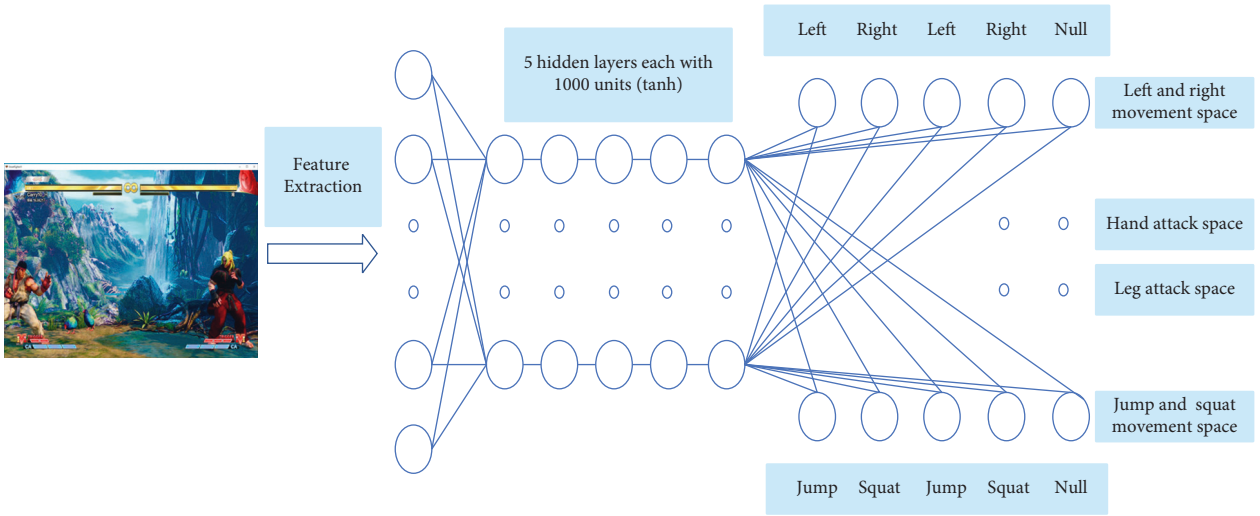

Figure 3: Network architecture of agent in Street Fighter V.

Table 1: List of hyperparameters.

Hyper parameter	Value
Mini-batch size	100
Replay memory size	40000
Target network update frequency (round)	4
Discount factor	0.95
Initial exploration	1.0
Final exploration	0.49
Image size	224

sufficient. However, based on the model we have changed, we should expand the experience pool.

(ii) Mini-batch size refers to the number of samples randomly selected for learning in the training process.

(iii) Target network update frequency (round) means that the target network is trained once after the specified number of rounds. The update frequency should not be too fast. It is necessary to ensure that there are enough training attempts for intelligent physical fitness before learning.

(iv) Initial exploration and final exploration, respectively, represent the exploration rate at the beginning of the training and the final exploration rate after the completion of the training. These two parameters are closely related to the agent's exploration behavior in the training process, corresponding to the model free method. It is often used to make agents break through the algorithm bottleneck and reach the next stage. A higher exploration rate means that the agent has more random choices and is not limited by the environment, but it also means that the number of decisions based on learning is less. The lower exploration rate is to enable the agent to make the best choice in line with the current situation at a certain time based on the existing training results after a certain stage of training, but at the same time, the possibility of breaking through the bottleneck is also limited. Therefore, the exploration rate is an extremely important parameter, which requires us to make a balance between "making the best decision" and "free exploration to break through the bottleneck." For this reason, we set the exploration rate that will slowly decrease and then stabilize with the increase of experience pool data during the training process.

(v) Image size refers to the size of each frame used for in-depth training and learning in the input system. We have tried other size image settings. However, too large image setting will not only reduce the efficiency of agent training and learning but also reduce the efficiency of making decisions based on the environment. The setting of too small pictures improves the efficiency of the school, but reduces the quality of learning, thereby affecting decision-making. Therefore, we use the setting of image size in most of the experiments in this paper.

Figure 4(a) shows the rewards for 250 rounds of training. As can be observed, the reward curve does not directly increase or decline as the training proceeds, but constantly fluctuates as the number of rounds of training rises. This indicates that the agent continuously learns. Eventually, the reward curve may approach a certain value.

Figure 4(e) shows the winning rate curves for 250 rounds of training. According to this figure, these curves keep fluctuating except in extreme situations. Such fluctuations are generally caused by the continuous learning and exploring of the agent, which is in a state of constant trials and errors. However, the curves may finally tend to be stable. In our opinion, these curves are normal and sound.

After training, three independent tests were performed by fighting against in-game AI of level 4 (level 1 is the lowest and the easiest and level 8 is the highest and the most difficult). In 10 rounds of fighting, the winning rate turns out to be 100%; moreover, it wins 27 of 30 rounds, generating a

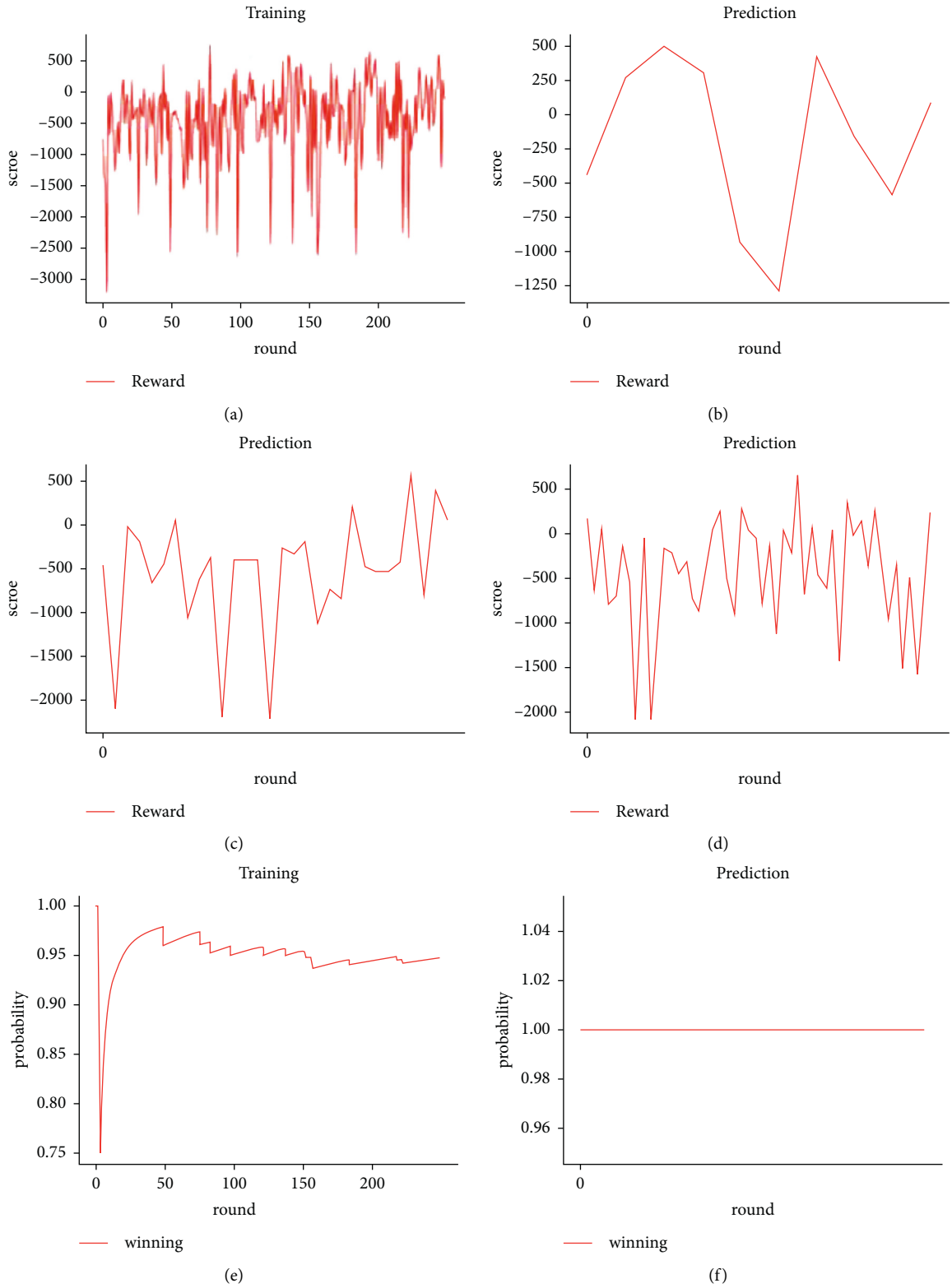

Figure 4: Continued.

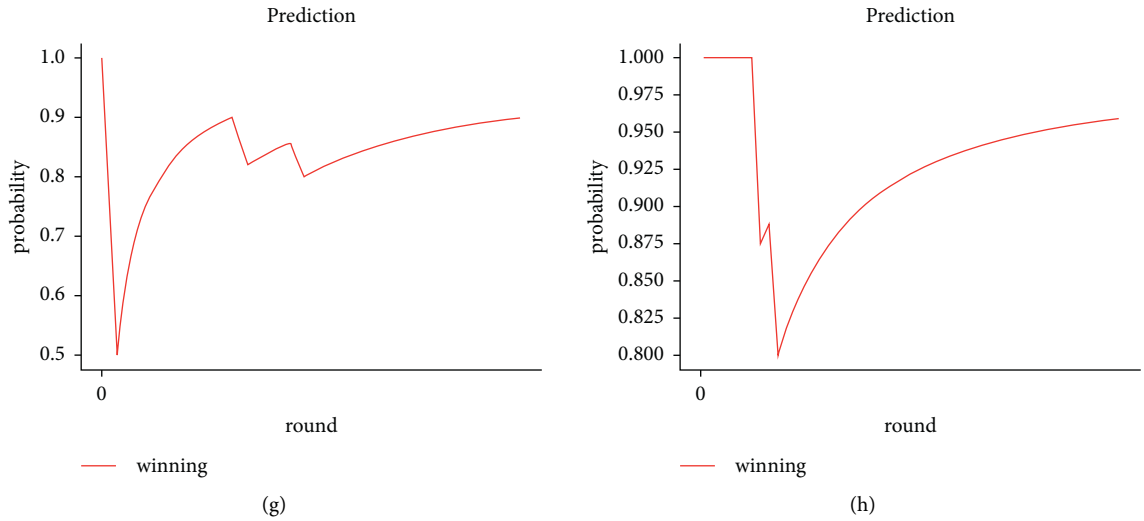

Figure 4: Training (a~h).

Table 2: Winning of rate in training/testing.

Training round	Rate (%)
250	95
Test round	
10	100
30	90
50	96

winning rate of 90%. After 50 rounds of fighting, it wins 48 rounds, with a winning rate of 96%, as shown in Table 2. Relevant reward curves (Figures 4(b)–4(d)) are consistent with winning rate curves (Figures 4(f)–4(h)) one by one. According to these figures, the trained AI can be endowed with more positive rewards during the test even if the reward curve fluctuates. Additionally, the winning rates of the trained AI are always superior to those during training because its decisions are gradually stable and efficient.

9. Conclusion and Future Work

There are also some interesting phenomena during the experiment. For instance, AI becomes increasingly inclined to select more robust playing methods. Instead of frequently jumping, it tends to squat or make other small movements. When it is rather far away from the rival, the AI may adopt a jump-based skillset, such as uppercut + mid-/long-range spin kick, so as to bridge the distance and make it convenient for attacking the rival. At a normal distance from the rival, a long-range hand or leg attack is selected by AI to protect itself from being hit and kill the rival at the same time. If the distance from the rival is rather short, AI may frequently makes short-range hand attacks that require much shorter animation time. Once the rival jumps, AI also attacks the rival in the air, such as emitting light waves or jumping to hit the rival.

This paper successfully enables agents to make various combinations of techniques in the case of only basic movement, jumping, and attack. This also shows that agents can gradually determine the optimal strategy in learning. At the same time, we have verified the intelligence of reinforcement learning algorithm DDQN in fighting games with a very high winning rate (nearly 95%).

For future works, efforts will be made to conduct more in-depth improvement of the agent structure and the entire system, enabling it to beat more advanced in-game AI (level 5 and above). It is also expected that the agent can effectively fight against advanced players before long.

Data Availability

The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research work was funded by the project FRG-22-005-INT and granted by the Research Fund of Macao University of Science and Technology (FRG-MUST).

References

[1] S. Yoon and K. Kim, "Deep Q networks for visual fighting game AI," in *Proceedings of the 2017 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 306–308, New York,NY,USA, August 2017.

[2] I. P. Pinto and L. R. Coutinho, "Hierarchical reinforcement learning with Monte Carlo tree search in computer fighting game," *IEEE Transactions on Games*, vol. 11, no. 3, pp. 290–295, 2019.

[3] Y. Takano, H. Inoue, R. Thawonmas, and T. Harada, "Self-Play for training general fighting game AI," in *Proceedings of the*

2019 Nicograph International (NicoInt), p. 120, Yangling, China, July 2019.

[4] Y. Takano, S. Ito, T. Harada, and R. Thawonmas, "Utilizing multiple agents for decision making in a fighting game," in *Proceedings of the 2018 IEEE 7th Global Conference on Consumer Electronics (GCCE)*, pp. 594-595, Nara, Japan, October 2018.

[5] H. Inoue, Y. Takano, R. Thawonmas, and T. Harada, "Verification of applying curiosity-driven to fighting game AI," in *Proceedings of the 2019 Nicograph International (NicoInt)*, p. 119, Yangling, China, July 2019.

[6] Y.-J. Li, H.-Y. Chang, Y.-J. Lin, P.-W. Wu, and Y.-C. F. Wang, "Deep reinforcement learning for playing 2.5D fighting games," in *Proceedings of the 2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 3778–3782, Athens,Greece, October 2018.

[7] R. Ishii, S. Ito, M. Ishihara, T. Harada, and R. Thawonmas, "Monte-carlo tree search implementation of fighting game AIs having personas," in *Proceedings of the 2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 1–8, Maastricht, Netherlands, August 2018.

[8] H. R. Bezerra, L. F. W. Góes, and A. R. da Silva, "Development of an autonomous agent based on reinforcement learning for a digital fighting game," in *Proceedings of the 2020 19th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pp. 1–7, Recife, Brazil, November 2020.

[9] D.-W. Kim, S. Park, and S.-I. Yang, "Mastering fighting game using deep reinforcement learning with self-play," in *Proceedings of the 2020 IEEE Conference on Games (CoG)*, pp. 576–583, Osaka,Japan, August 2020.

[10] I. Oh, S. Rho, S. Moon, S. Son, H. Lee, and J. Chung, "Creating pro-level AI for a real-time fighting game using deep reinforcement learning," *IEEE Transactions on Games*, vol. 14, no. 2, pp. 212–220, 2022.