Two Multi-Unit Artificial Intelligence Decision Modes in Video Games: A Comparison

Qisen Lin

School of Information Engineering, Wuhan University of Technology, Wuhan Hubei 430070, P. R. China

ABSTRACT

Using AI to manipulate game characters is already common and has been applied to various types of games. However, many current studies focus on AI operating a single role. In future online games, there will be a great demand to operate multiple units simultaneously for collaborative movement, and similar needs have already appeared in current MMORPG. In popular battle games nowadays, there is often a game mode where multiple players belonging to multiple factions engage in battles with each other. If AI needs to be deployed, multiple units need to be operated simultaneously to engage in combat. In response to this issue, two network architectures were proposed and trained and tested in a multiplayer game simulation environment. By comparing the training effects, one of the network structures known as "octopus- shape" is considered to be more effective and valuable for development. This result will contribute to the development of multi unit collaborative artificial intelligence in the future, helping to establish true AI games and real-world automated management systems.

KEYWORDS

Artificial intelligence; Reinforcement; Online game; Neural network; Cooperative system

DOI: 10.47297/taposatWSP2633-456909.20230402

1 Introduction

Compared to the 1958 film "*Tennisfortow*", after more than half a century of development, gaming technology has undergone a huge leap, and the number of players in the game has also rapidly increased. In the 20th century, many electronic games faced only one to two live players and a few non player characters based on simple logic control. Nowadays, popular multiplayer electronic games have more players and more complex gaming environments. For current players, they prefer to interact with more intelligent and challenging game characters compared to game characters that run based on simple logic. At the same time, traditional simple logic controlled NPC roles are also difficult to meet the increasing demand for realism. On the other hand, the scale of online games continues to expand, and the number of online players continues to grow. In order to provide a good gaming environment and maintain the daily operation of games, it is also necessary to use automated artificial intelligence to alleviate the work pressure of human workers. In order to achieve a game environment that is more realistic and in line with player needs, further research is needed on AI investment. In addition, electronic games themselves can be seen as a simplified virtual environment compared to the real world[1]. Research on the investment of AI technology in electronic games can also provide reference for AI research in the real world. So far, AI technology has been applied in the gaming

industry and other management industries, such as port management, proving the feasibility of using AI technology for decision-making and management.

There have been many research results on game AI using deep reinforcement learning. AI has proven the possibility of surpassing human performance in *Adali* games[2], and can also achieve levels similar to human players in more complex modern electronic games such as Shooting Games[3] and Combat Games[4]. However, currently many games using artificial intelligence still focus on operating a single game unit (playing player or non player roles), and in popular electronic games, especially Online Games[5], there are often plural player and plural non player roles, which requires AI to be able to operate multiple units simultaneously. For the subsequent development of AI, it is necessary to conduct preliminary research on its network architecture and clarify the direction of future development. In response to this issue, this article proposes two feasible multi unit artificial intelligence decision-making models and their network architectures, and tests their advantages and disadvantages in research. To this end, a multiplayer collaborative game will be designed, and two AI models will be trained and tested in this environment based on reinforcement learning methods to evaluate their learning rate and effectiveness, and explore feasible Multi Unit Artificial Intelligence Decision Modes.

2 Materials and Methods

(1) Basic theory

1) Reinforcement

Reinforcement Learning is a widely used technology[6].Reinforcement Learning is a form of machine learning. In the process of reinforcement learning, the agent continuously interacts with the environment to obtain the mapping between the environment and behavior, thereby achieving correct decision-making. In order to seek the optimal strategy, the intelligent experience constantly tries and makes mistakes, judges the quality of actions based on the feedback from the environment, and tends to choose the optimal strategy.

2) Q-Learning[7]

Learning algorithm is a common algorithm in reinforcement learning. Unlike previous Monte Carlo reinforcement learning, Q-Learning does not require the use of a complete trajectory for learning and can achieve more efficient learning. The core of Q-Learning is a $Q(S,a)$ function that selects corresponding actions based on the values calculated for different actions in the environment. By calculating the Q value for each state and establishing a Q-table, it is easy to find a suitable decision path to select actions. In order to achieve better decision-making, it is necessary to obtain better values through multiple attempts and learning. Due to the difficulty in testing all possibilities in real environments, Q-Learning updates values with a limited number of tests.

The Q-value update method for Q-Learning is as follows:

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha(R_s^a + \gamma Q_t(s_2,a_2) - Q_t(s,a)) \quad (1)$$

Among them, $Q_{t+1}(s,a)$ is the Q value of environment s and action a after the t+1 update, $Q_t(s,a)$ is the Q value after the t update, α is the learning rate of the update, γ is the reward decay factor, $Q_t(s_2,a_2)$ is the Q value of the environment s_2 after executing the action a_2, and R_s^a is the reward that can be immediately obtained by executing action a in environment s. By calculating the current

Q-value and the next Q-value, it is possible to update the Q-value, which greatly improves efficiency compared to previous algorithms.

3) Deep Q-Learning[2]

Due to the large number of states in actual environments, it is difficult to establish a complete Q-table. Therefore, neural network and deep learning technologies were introduced into the reinforcement, which is called Deep Q-Learning, where the network used is called Deep Q-Network (DQN). With the help of neural network, it is possible to fit the characteristics of complex functions, and through continuous learning, it can simulate ideal $Q(S,a)$ functions and approximate the effect of Q-Learning.

During training, DQN takes start state as input and outputs Q-values for all actions, selecting actions based on Q-values. After an Action is input into the environment (usually a simulator), the environment will change to generate a new state and the reward obtained by executing the Action. By obtaining the past state, executed actions, rewards, and new states, neural networks can be trained based on these data. When training the neural network, supervised learning is used to update parameters by calculating the Loss-value between the ideal Q-value: Q-target and the actual Q-value: Q-real.

According to the principle of Q-learning, the calculation methods for Q-real and Q-target are as follows:

$$Q_{\text{target}} = \text{reward} + \gamma * \max\left[Q\left(s_{\text{new}},\ a_{\text{new}}\right)\right] \quad (2)$$

$$Q_{\text{real}} = Q(s_{now}, a_{now}) \quad (3)$$

Among them, $Q(s_{now}, a_{now})$ is the Q-value obtained by selecting the action a_{now} for the current neural network in the environment s_{now}, the reward from the environment feedback, and γ is the reward decay factor that reflects the value ratio of future rewards at the current time, $\max\left[Q\left(s_{\text{new}},\ a_{\text{new}}\right)\right]$ is the maximum Q-value obtained by processing the neural network in the new environment s_{now} after executing the action a_{now}. After obtaining Q-target and Q-real, supervised learning methods can be used to calculate the loss-value using the loss function and update parameters using Gradient Descent.

(2) Experimental environment

1) Visual environment

The visual environment "SVS" used in the experiment is a self-designed multiplayer collaborative gaming environment, which is a "Two Camp Multiple Player" gaming mode. The game unit is divided into two camps: "Anti Submarine Ship" and "Submarine", which are set at both ends of the chessboard at the beginning of the game. A chessboard is a 32 * 32 matrix that uses matrix elements to record the positions of each unit. Submarines operated by built-in logic need to cross the entire chessboard to reach the side line at the other end, while anti submarine ships operated by AI need to sink them before reaching the side line. In the experiment, three anti-submarine ships and two submarines will be dropped onto the side lines on both sides of the chessboard. The initial position of the anti-submarine ship is fixed, while the position of the submarine is random. In order to train AI's ability to face multiplayer games, AI units can only move once in a turn, while submarine units can move twice. This means that AI without coordination will find it difficult to achieve game tasks. In the game, the chessboard is checked multiple times to confirm if any submarines have arrived at the side line or been sunk.

2) Neural network

The Neural Network is the core of this experiment. The Neural Network will calculate Q-values for different actions based on the input environments, and then select the action corresponding to the maximum value in a set of Q-values as the output. In order to achieve better training results, two network architectures were designed for experiments, namely "Hand-Shape" and "Octopus-Shape".

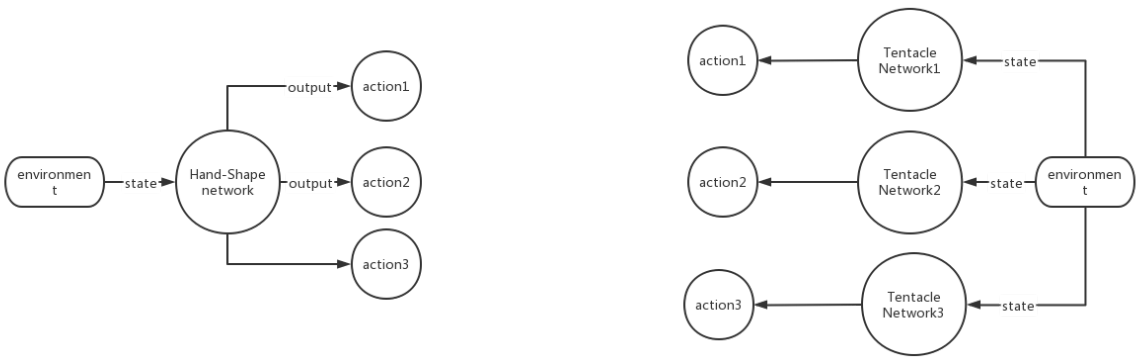

Fig.1. Hand-Shape and Octopus-Shape

Hand-Shape takes state as input from the environment and outputs 12 Q-values, corresponding to four actions that can be executed in three units. These 12 Q-values are divided into three groups, and the action corresponding to the maximum value in each group is the action selected by the corresponding unit. When updating parameters, each set of Q-reals consists of the Q-values corresponding to the actions selected by three units to form a three-dimensional vector, while Q-targets are composed of individual Q-targets for each unit to form a three-dimensional vector.

$$Q_{target} = [(reward + \gamma * Q(s_{new}, a^1_{new}), (reward + \gamma * Q(s_{new}, a^2_{new}), (reward + \gamma * Q(s_{new}, a^3_{new})] \quad (3)$$

$$Q_{real} = \left[\left(Q(s_{now}, a^1_{now})\right), \left(Q(s_{now}, a^2_{now})\right), \left(Q(s_{now}, a^3_{now})\right)\right] \quad (4)$$

Octopus-Shape has three neural networks with identical network structures as subnetworks, which are named as 'Tentacle Network ' Each Tentacle Network takes the same state as input, generating Q-values for four types of actions, and selecting the action corresponding to the maximum value as the instruction for the corresponding unit. When updating parameters, each Tentacle Network calculates Q-target and Q-real separately, using the same method as regular Deep Q-learning.

To control variables, the tension network of Hand-Shape and Octopus-Shape will use the same Convolutional Neural Network[8] as the network structure. This network structure has three convolutional layers and three fully connected layers, and the output is determined based on the network type of the application.

3) Experimental methods

The purpose of the experiment is to compare the advantages and disadvantages of two network structures, that is, to determine the training frequency and task completion effect required for the two network structures to achieve tasks. For this purpose, the "SVS" game will be used as a simulation environment for training. The training process will be divided into two steps: observation period and training period. During the observation period, the program will constantly input the game map as a state into the Neural Network, and determine the corresponding actions based on its output. During this period, in order to improve training effectiveness, there is a certain probability of selecting actions that correspond to non maximum values to increase the diversity of experience. After the action is completed, the past action state, post action state, action, and feedback reward will be recorded

in the database for future training calls. The capacity of the database is 2000 sets of data. Once the capacity is exceeded, the new data will overwrite the old data. During the observation period, due to the accumulation of data, the Neural Network will not be updated. After the observation period ends, the training period begins. During the training period, the steps for playing games and accumulating data are exactly the same, except that loss-value calculations and network parameter updates will be carried out during the training period.

The game has a scoring standard used to provide rewards for training. Considering the difficulty of the game, as long as AI can operate an anti submarine ship to sink one or more submarines, it can be judged as a game victory; If both submarines reach the side line, the game will be judged negative. In addition, in order to prevent the game from continuing endlessly, when the number of rounds exceeds 60, the game will be forced to end. If no submarines are sunk at this time, the game will be judged negative.

Table 1. End of Game Reward rule

Seeking submarine	Side-line submarine	Submarine On-map	Out-time	Reward
2	0	0	No	1
1	1	0	No	0.5
0	2	0	No	-1
1	0	1	Yes	0.5
0	1	1	Yes	-1
0	0	2	Yes	-1

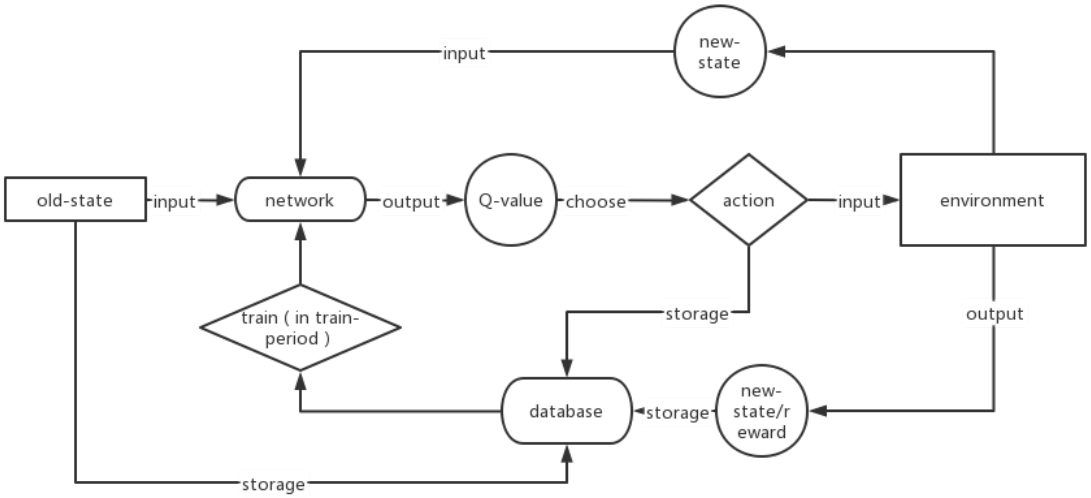

Figure 2. Experiment flow chat

In training, 40 randomly selected states are extracted from the database, and the Q-real and Q-target corresponding to each data are calculated based on their data. They are combined into Q-real vectors $\left[Q^1_{real}, Q^2_{real}\cdots, Q^{40}_{real}\right]$ and Q-target vectors $\left[Q^1_{\text{target}}, Q^2_{target}\cdots, Q^{40}_{target}\right]$, respectively. Then, the MSE (Mean Square Error) function is used to calculate the loss-value, and the network parameters are updated using Back-propagation method.

3 Results and Discussions

(1) Data

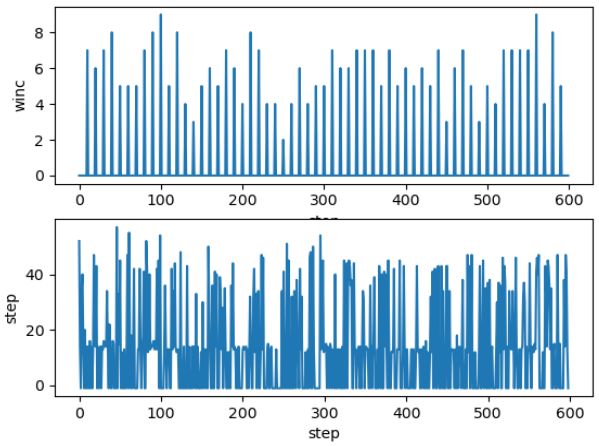

Figure 3. Octopus-Shape data-1

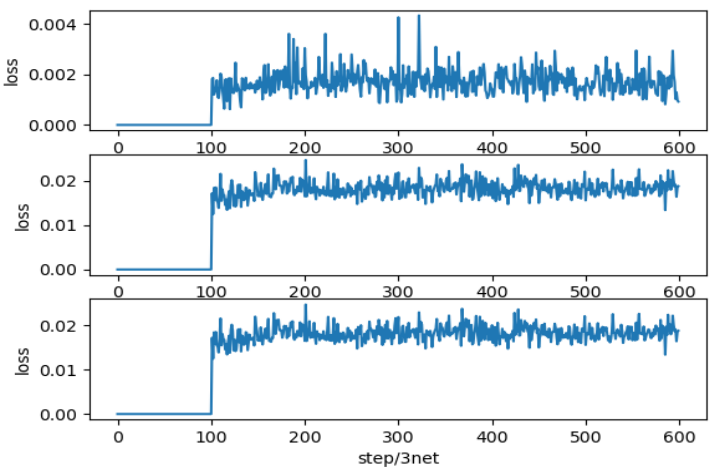

Figure 4. Octopus-Shape data-2

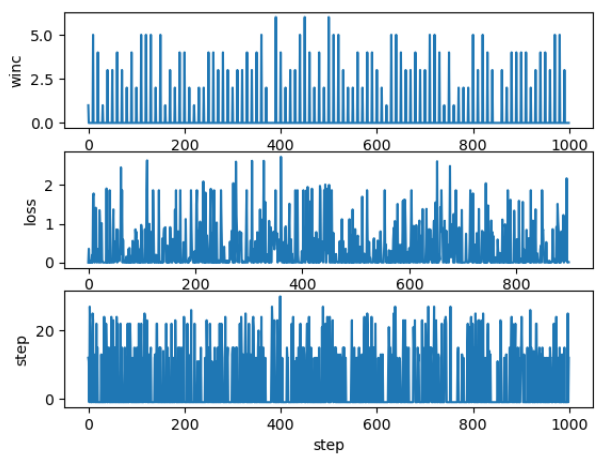

Figure 5. Hand-Shape data

The measured data consists of three parts. 'Winc' refers to the increase in the number of successful games after every 10 games of training, used to display the growth rate of game success rate. 'loss' refers to the change in loss-value during each training session, used to display the difference between Q-value and Q-target. The larger the loss-value, the less ideal the parameters of the neural network are. The 'winstep' is used to display the number of rounds consumed for each game victory, and is an indicator used to assist in evaluating the quality of the neural network.

(2) Experimental data analysis

The experimental data shows the number of task completion times per ten games, the loss-values after each training, and the number of steps required to complete each task. According to the experimental results, when the training frequency reaches 600, the Octopus-Shape can achieve an overall task completion rate of around 50% to 60%, while the Hand-Shape needs to be trained 1000 times to achieve a completion rate of nearly 30%. Meanwhile, during the completion stage of the training (i.e. the last 100 games at the end of the training period), Octopus-Shape can achieve a 60% accuracy rate, while Hand-Shape can only achieve about 20% accuracy. From this, it can be concluded that compared to Hand-Shape, Octopus-Shape is more suitable for completing tasks with multiple agents collaborating.

However, it can be seen that there are still certain shortcomings in the training effect, and there is still room for improvement in the task completion rate. It can be seen that in many missions, the goal of completing the game has been achieved, but the time consumed often reaches over 40 rounds. After training, there is also a certain probability that the mission cannot be completed. To improve this defect, two approaches can be taken. On the one hand, the number of training sessions can continue to increase, and the experience database can also be expanded. On the other hand, the structure of the tension network that constitutes the Octopus-Shape can continue to be optimized, including increasing its depth or choosing other network structures.

4 Conclusion

This article investigates the applicability of two neural network structures: Hand-Shape and Octopus-Shape in multi unit collaborative tasks, and tests the two network structures using simulated games as experimental environments. The test results indicate that the Octopus-Shape configuration has better learning efficiency and greater development potential compared to the Hand-Shape. This conclusion will contribute to the further development of AI for online games and even complex real-world tasks, such as traffic scheduling AI, facility management AI, or enterprise management AI. In addition, this conclusion also puts forward new requirements for specific AI design. In order to increase the completion of tasks, it is necessary to further research and design the algorithm, including improving the training algorithm and improving the subnet structure.

About the Author

Qisen Lin is undergraduate of Wuhan University of Technology, and his research field is information engineering.

References

[1] Q. Fu, C. Fan, Y. Song and X. Guo. (2020) .Alpha C2–An Intelligent Air Defense Commander Independent of Human Decision-Making. IEEE Access, vol. 8, pp. 87504-87516.

[2] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.A. (2013). Playing Atari with Deep Reinforcement Learning. *ArXiv, abs/1312.5602*.

[3] I. Kodaka and F. Saitoh. (2021). A Study on Application of Curriculum Learning in Deep Reinforcement Learning: Action Acquisition in Shooting Game AI as Example. In: 2021 IEEE 12th International Workshop on Computational Intelligence and Applications (IWCIA), Hiroshima, Japan, pp. 1-6.

[4] S. Yoon and K. Kim. (2017) .Deep Q Networks for Visual Fighting Game AI. In: 2017 IEEE Conference on Computational Intelligence and Games (CIG), New York, USA, pp. 306-08.

[5] Gandolfi, E., Ferdig, R. E., & Soyturk, I. (2023). Exploring the Learning Potential of Online Gaming Communities: An Application of the Game Communities of Inquiry Scale. New Media & Society, 25(6), 1374-93.

[6] Y. Chao, J. Liu, S. Nemati, and G. Yin. (2021). Reinforcement Learning in Healthcare: A Survey. ACM Comput. Surv., 55(1): Article 5.

[7] Clifton, J. Laber, E. (2020). Q-Learning: Theory and Applications. Annual Review of Statistics and Its Application, 7(1): 279-301.

[8] Z. Li, F. Liu, W. Yang, S. Peng and J. Zhou. (2022) .A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. IEEE Transactions on Neural Networks and Learning Systems, 33(12): 6999-7019.