Aritificial King of Figter: Fighter game agent based on deep reinforcement learning.

Xiaoqing Zhang, Xi Lin, Boyong He, Yuebing He, Haoyin Huang

36920221153149, 36920220156885, 36920220156889, 36920220156890, 36920220156884
Institute of Artificial Intelligence, Xiamen University, China

Abstract

In 1997, SNK released an arcade fighting game called KOF97. With the breakthrough and development of Deep Reinforcement Learning (DRL), it is possible to train AI to play arcade fighting games. In order to achieve the goal of AI winning in the King of Fighters, we propose to use the deep reinforcement learning model to train the characters in the game, in order to achieve smooth combo of different characters.Furthermoreto realize the purpose of AI defeating the random release skill model and even defeating human operations. Start the game based on the MAME simulator environment, and establish the necessary environment for model training. Based on the process oriented reward mechanism, the role can make skill use decisions more quickly and smoothly, so as to defeat human operations in terms of reaction speed and skill use intensity. Our experiment proves that the performance of the characters trained by our model really exceeds that of the random skill release model, which shows that the model really enables the characters to learn to play games. Due to the limitation of computation ability and time consuming, our model can only train three characters, but our work has verified that deep reinforcement learning has a strong ability in game training.

Introduction

Electronic games have been a hot issue since the 21st century. With the development of computer science and technology, more complex games spring up. However, no matter how attractive these new games have been, they would never exceed their predecessors, i.e. the first generation of electronic games: Super Mario, The King of Fighters, Battle City, Tetris.... When the electronic games first created, nothing can be used for reference, these classic games became the founder of the new world of electronic games. Decades have passed but today's new games are still drawing lessons from the original ideas of old games. Moreover, we spent our younger ages along with these old games, which contain our memories of youth and cronies, and this is why our team select the topic of playing electronic games with machine learning methods.

Figure 1: Learning results of the model. In the training and learning of 15000simulator steps training, the model learned the corresponding basic combo for different roles under the condition of reducing training loss by setting process oriented reward mechanism, which demonstrated the effectiveness of the model.

In 1997, SNK released an arcade fighting game called The King of Fighters' 97 (KOF97). The hearty combo in the game, the hot blooded Kyo Kusanagi, the rebellious Yagami, and Mai Shiranui that provoked many ignorant teenagers finally created the classic of the times. It also buried a warm heart to win the title of the [King of Fighters] for every boy who played KOF '97. With the breakthrough and development of Deep Reinforcement Learning (DRL), it is possible to train AI to play arcade fighting games. In order to live up to the heart of every hot blooded teenager, and to further explore the charm of deep reinforcement learning, our team plans to use a variety of deep reinforcement learning technologies, take the arcade fighting game Boxing 97 as the background, and achieve the goal of AI winning [King of Fighters], horizontally compare the advantages and disadvantages of different depth reinforcement learning networks, and explore the impact of different manual strategies on AI's ability to use combos and combat. At the same time, it is proposed to explore the learning process of AI from different visual angles, and further analyze the role of exploration networks and strategies in the training process.

Since AlphaGo (Silver et al. 2016) defeated the human professional champion Go player with a score of 4:1, AI has gradually moved from perceptual intelligence to

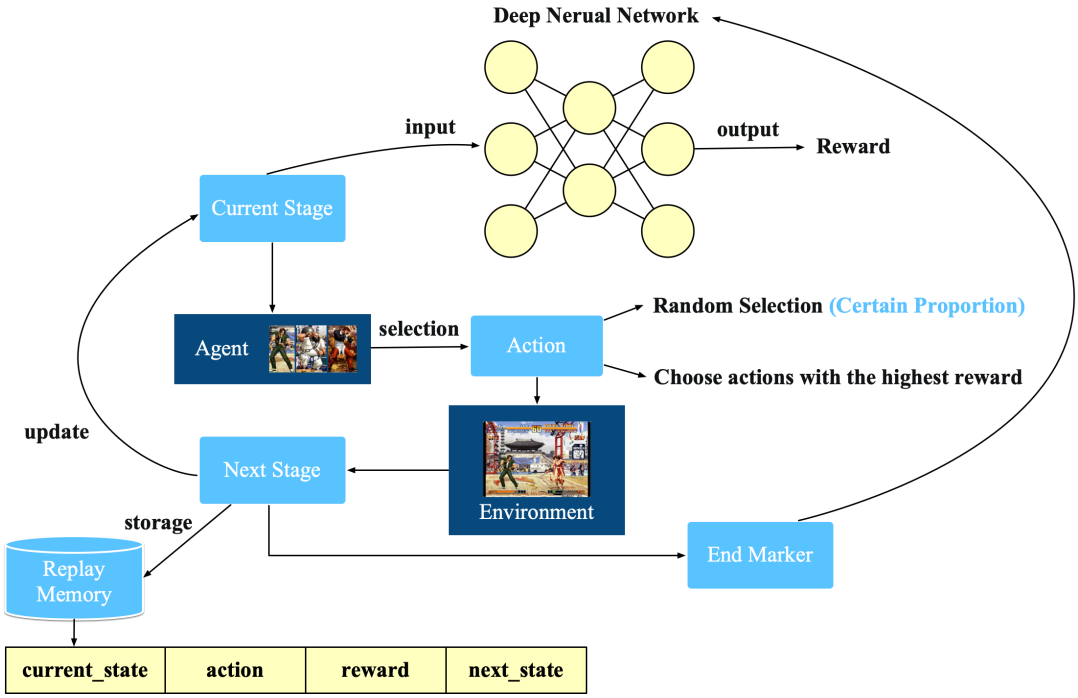

Figure 2: Our model architecture. First, the agent makes the character randomly select at a certain proportion or execute the action with the highest reward according to the current state of the screen and the character's health, the number of winning fields, relative position, energy, defense state and other information, and then waits for the environment to respond. In the training stage, the reward is calculated by using the output of neural network in a weighted way.

decision-making intelligence. The breakthrough and development of its core technology, Deep Reinforcement Learning (DRL), has made it shine in the fields of bioinformation (Senior et al. 2020; Jumper et al. 2021; Tunyasuvunakool et al. 2021) , transportation (Tang et al. 2019; Qin, Tang, and Ye 2019) , automatic driving (O'Kelly et al. 2018; Li et al. 2019) , security (Jun et al. 2018; Havens, Jiang, and Sarkar 2018) and games, which also makes it possible for teenagers to realize their dreams. Therefore, our team plans to use a variety of in-depth reinforcement learning technologies, take the arcade fighting game The King of Fighters 97 as the background, and achieve the goal of AI winning [King of Fighters], horizontally compare the advantages and disadvantages of different in-depth reinforcement learning networks, and explore the impact of different manual strategies on AI's ability to use combos and combat. At the same time, it is proposed to explore the learning process of AI from different visual angles, further analyze and explore the role played by the network and strategies in the training process, and provide more scientific theoretical support for young people to realize their dreams and win the title of [King of Fighters].

Related work

Deep reinforcement learning algorithms. In recent years, deep reinforcement learning methods have achieved remarkable results in both turn-based and real-time games, and become the highest level of AI representation in many games. Deep reinforcement learning effectively combines the respective advantages of deep neural networks and reinforcement learning for solving end-to-end sequential decision optimization problems of intelligences in high-dimensional state spaces. DQN (Mnih et al. 2013) is the first end-to-end deep reinforcement learning algorithm. Since it was designed for video games, it uses a convolutional neural network to learn the state action value function Q. Specifically, the input of the network is a sequence of processed images, and after the convolutional network operation, the output is the value corresponding to all actions, thus realizing the function of Q. Dueling DQN (Freitas et al. 2016) and Double DQN (Van Hasselt, Guez, and Silver 2016) are two variants of DQN, where Double DQN is based on a mutually supervised idea of using two Q networks of the same structure with deviating parameters for evaluation to solve the problem of large Q values in DQN. The Dueling DQN network architecture is an idea to improve the DQN algorithm by optimizing the network structure.

Application of deep reinforcement learning in games. Deep reinforcement learning, as one of the hot research directions in recent years, has received a lot of attention due to its excellent performance on Go and other video games. Unlike traditional deep learning, deep reinforcement learning allows an intelligent body to explore in the environment to learn strategies without labeled samples. For example, AlphaGo Zero (Silver et al. 2017) developed by DeepMind

completely abused humans on Go without using any human Go data; Dota Five (Berner et al. 2019) developed by OpenAI reached the top level of human players on DOTA game; AlphaStar (Vinyals et al. 2019) developed by DeepMind also defeated human professional players in StarCraft. These have become milestones in the development of deep reinforcement learning, proving the power of deep reinforcement learning in gaming scenarios.

Real-time fighting games are a challenging and enjoyable real-time game problem. It requires an intelligent body to make an effective choice from a large set of candidate moves in a very short reaction time. Real-time fighting games are usually played in a 1 vs. 1 format, with a fixed initial blood value and zero initial energy value, so that the opponent can be effectively hit to gain energy value and take high damage actions according to the accumulated energy value, and finally defeat the opponent by the blood difference. According to the action space dimension, fighting games can be divided into two types: 2-dimensional space and 3-dimensional space. Real-time fighting games interact with the environment through trial-and-error reinforcement learning, effectively balancing the relationship between exploration and exploitation in the unknown environment of the model, and learning the optimal strategy by maximizing the reward signal obtained from cumulative sampling.

The data triples for real time fighting games are state, action and reward signals. The game engine provides the input state information to the model, either as a one-dimensional physical value or as a two-dimensional game screen. Fighting game states include: character attributes, skill attributes, distance attributes, and time attributes. The character attributes are blood, energy, position, speed, movement, character state (e.g. standing, crouching, falling, empty) and remaining action frames; the skill attributes are energy consumption, skill damage and skill attack attributes (e.g. close attack or long range); the distance attributes are the relative physical distance and position of the two sides; and the time attributes are the remaining game time or frame count. The action space is discretized and represented as a set of candidate actions in terms of executable actions. The reward signal is used as a guide to motivate the intelligent model to master the fighting strategy by using the final win/lose signal, the blood difference and the step penalty as reward guides. The specific inference and optimization process of deep reinforcement learning is shown in Figure 1. Thus, the deep reinforcement learning method can be directly adapted to the fighting game task solving process.

Application of reinforcement learning in open world games. Open-world games are a type of roaming game level design in which players are free to roam around a virtual world, freely choosing when and how to complete the game tasks, and constantly opening up new maps. Currently the more popular open-world games are the Genshin Impact developed by Mihayo. The game character generation and character animation cannot be generated without deep reinforcement learning methods.

In the Genshin Impact game there are different main quests as well as linear quests, and in each quest there are a large number of NPCs to help complete the game's plot increasing the challenge of the game and making the game's world more realistic. Deep reinforcement learning is to let the intelligence explore the environment continuously and learn by trial and error to maximize the accumulated rewards to get the optimal strategy, which means it does not need to have ready-made labeled samples or artificially written rules, so using deep reinforcement learning to game NPCs would be a worthy solution to consider.

To apply reinforcement learning in the Genshin Impact's game scenario, the first thing to do is to abstract the scene as an MDP. Taking the NPC to be trained as the intelligence, the game it faces as the environment, and the character gets its current state and rewards from the game and makes the corresponding actions. The state is used to represent the feature information of the current character, either directly from the raw information of the current game screen, which is then encoded by techniques such as CNN to extract features, or semantic information extracted through the game API. Similarly, the character's action can be the same keystroke information as the human player, or it can be a direct control of a higher level of action API. The reward part, where different rewards imply different learning goals, needs to be designed according to the goals and want to achieve. In simpler games, the reward can be directly the score over time or even the win/loss of a game, while in more complex games, such a simple setup may not be trainable or converge slowly. This part is also known as reward engineering in reinforcement learning applications After abstracting the game as an MDP, it can choose the appropriate reinforcement learning algorithm for training, such as DQN algorithm, PPO algorithm, etc. Similar to other machine learning methods, the hyperparameters, network structure, and even the learning algorithm inside these algorithms affect the learning effect and need to be adjusted according to the training effect, as well as the reward part needs to be re-optimized and designed according to the effect.

The characters in the Genshin Impact come to life with a very smooth and organic movement effect. To achieve this effect, there are various solutions, from the earliest sprite animation and rigid stratum animation to today's masked animation, and the rendering is enhanced step by step, but it is difficult to simulate the diversity of behaviors exhibited in the real world, and it is often labor-intensive to design the control logic manually and difficult to adapt to new scenarios and situations. To address this problem, reinforcement learning offers a possible idea for motion synthesis, in which an intelligent body learns by trial and error in repeatedly performing actions of various skills to reduce the heavy reliance on human labor. However, simply allowing an intelligent body to explore freely may produce actions that do not affect the skill goal but are meaningless, such as irrelevant upper body movements, awkward postures, etc. While the movements can be made more natural by incorporating more realistic bioengineering models, it is very difficult to construct high-fidelity models and the resulting movements are likely to remain unnatural. Therefore, an ideal system for automatically generating character animations should first provide the intelligence with a set of reference actions that

meet the requirements, and then use them as a basis for generating behavioral actions that match the target and are physically realistic. For example, attempts to use DeepLoco, which uses deep reinforcement learning methods to imitate motion data, and Generative Adversarial Imitation Learning (GAIL) methods to generate actions have yielded good results.

Methods

DQN is a reinforcement learning algorithm based on Q learning and deep learning. It is able to compare the expected utility of the available operations (for a given state) without the need for an environment model. Also it can handle stochastic transition and reward problems without the need for tuning. It has now been shown that for any finite MDP, Q learning eventually finds an optimal policy where the expected value of the total reward return is maximally achievable for all successive steps starting from the current state. Before learning starts, Q is initialized to a possible arbitrary fixed value (chosen by the programmer). Then at each time t, the agent chooses an action a_t, then gets a reward R_t, and enters a new state S_{t+1} and the Q value is updated. At its core is the value function iteration process, as:

$$\begin{aligned} &Q\left(s_t, a_t\right) \leftarrow Q\left(s_t, a_t\right) \\ &+\alpha \cdot\left[r_t+\gamma \max_{\pi} Q\left(s_{t+1}, a_t\right)-Q\left(s_t, a_t\right)\right] \end{aligned} \tag{1}$$

where α is the learning rate, γ is the discount factor. First initialize the value function matrix, start the episode, then select a state state, while the intelligence selects the action according to its own greedy strategy. After the intelligence will apply the action to get a reward R and S', calculate the value function, and continue to iterate the next process.

The reward r is defined as the change in the game score after taking action a at the current time t, state s is taken as r_t, as:

$$r_t=\left\{\begin{array}{cc} 1 & \text{increase} \\ 0 & \text{noexchange} \\ -1 & \text{decrese} \end{array}\right. \tag{2}$$

And long-term cumulative discount rewards R_t are then defined as:

$$R_t=\sum_{k=0}^{T} \gamma^k r_{t+k+1} \tag{3}$$

Therefore the core of the whole process becomes how to determine to approximate the value θ of the function. The most classical approach is to use gradient descent minimization loss function to continuously debug the network weights. Loss function is defined as:

$$\begin{aligned} &L_i\left(\theta_i\right)=E_{\left(s, a, r, s^i\right) \sim U(D)} \\ &\left[\left(r+\gamma \max_{a'} Q\left(s', a^i ; \theta_i^-\right)-Q\left(s, a ; \theta_i\right)\right)^2\right] \end{aligned} \tag{4}$$

where θ_i^- is the target network parameter for the i_{th} iteration and θ is the Q-network parameter. The next step is to find the gradient, as:

$$\begin{aligned} &\frac{\partial L_i\left(\theta_i\right)}{\partial \theta_i}=E_{\left(s, a, r, s^i\right) \sim U(D)} \\ &\left[\left(r+\gamma \max_{a'} \hat{Q}\left(s', a^i ; \theta_i^-\right)-Q\left(s, a ; \theta_i\right)\right) \nabla_{\theta_i} Q\left(s, a ; \theta_i\right)\right] \end{aligned} \tag{5}$$

Proposed solution

In this paper, the MAME simulator environment is first built to start the arcade game King of Fight 97. Meanwhile, the debugger mode of MAME is used to search for the screen, health, number of wins, relative position of the relevant characters, as well as the number of seconds, milliseconds and other related static memory addresses during the game. This is used to calculate the reward value and determine the combat status to control the further action of the game. At the same time, the open source framework MAMEToolkit is used to control the MAME simulator, so as to realize the game starting, coin slot, role control and other operations, and finally build the necessary Environment for reinforcement learning.

Among them, the LUA engine port used to operate the MAME simulator and the corresponding Field are respectively forward move, backward move, squat, jump, light punch, light leg, heavy punch, heavy leg, menu bar, coin slot and so on to control the game frame by frame. The final required memory address includes the character's health, number of wins, relative position, energy, defense status, etc.

Specifically, we first use the game minute hand, millisecond and health to determine whether it is a battle state. During the battle, we need to obtain the corresponding frame number and related memory address to input the network to calculate the relevant reward value. The number of wins and battle status of the character can be used to determine whether this is the end of a certain round, whether to enter the next stage of the battle, or whether the game is over and the game needs to be restarted.

The detailed reinforcement learning flow chart is shown in Figure 2. First, the Agent makes the role randomly select or execute the action with the highest reward according to a certain proportion of the information of the current state and the character's health, number of wins, relative position, energy, defense state, etc. Then the environment responds and calculates the reward. In our earlier version, The environment returns the difference between our lives and the lives of the other as a reward. After that, due to the execution of the action, the status of the characters in the environment changes, and reinforcement learning enters the next step. Before proceeding to the next round of decision process, the new state is also captured and combined with the previous information to form a memory data, which contains State(t), Action, Reward, State(t+1).

Experiment

Firstly, we selected three characters (Cao Zhanjing, Chen Guohan and Iori) for a total of 15,000 simulator steps training. The network learning rate was 0.005, the Agent's initial exploration rate was 0.9, and the exploration rate decay was

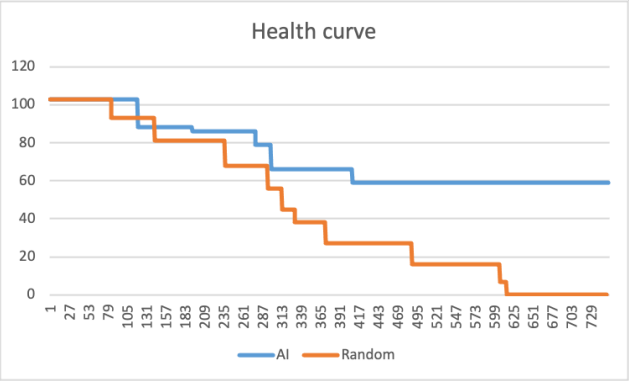

Figure 3: the health curve of the proposed method versus the random AI in the first round

0.9. Online learning was conducted every 500 samples. We want to maximize the reward value during training, and the reason the reward system is not based on winning is because we consider that the reward value is delayed, making training more difficult and time-consuming. Figure 3 shows the health curve of the proposed method versus the random AI in one experiential test process.

According to the observation of the experiment, we also found that the weight of the direction control press is different from that of the strike control. The direction control is only effective for one frame and has little influence in the game. However, the strike control, once pressed, lasts for a number of frames and has a significant impact in the game. For example, it takes many frames to complete the action of boxing. This means that an action taken in one frame will continue for many frames.

In addition, while the strike press is important compared to the directional control press, it requires a correspondingly more frequent press to be effective. To complete this action, and to humanize the AI action, we had the AI press the button repeatedly for 20 frames (1/3 of a second) before moving on to the next action. In other words, we're asking the AI to take actions and observe results in 1/3 of a second of play time, rather than per frame. According to the decision of AI, it can be found that AI has learned some combo moves in the training process, including forward charge, backward defense, jump kick, energy placement and so on.

Conclusion

In order to improve the ability of AI Boxing Emperor, our team members began to learn reinforcement learning from zero learning, built the Boxing Emperor environment from zero, and tried to build three reinforcement learning frameworks, including DQN, DDQN and Dueling DQN, and finally chose the original DQN model. But there is still a lot of room for improvement in the early versions of the current model, for two reasons. First, our team did not model the consistent movement of the characters, which is the key to the Emperor's victory. Second, the setting of the environment still has some deficiencies. For these two problems, we will improve later.

References

Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Debiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; et al. 2019. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*.

Freitas, N. D.; Lanctot, M.; Hasselt, H. V.; Hessel, M.; Schaul, T.; and Wang, Z. 2016. Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning*. JMLR.org.

Havens, A.; Jiang, Z.; and Sarkar, S. 2018. Online robust policy learning in the presence of unknown adversaries. *Advances in neural information processing systems* 31.

Jumper, J.; Evans, R.; Pritzel, A.; Green, T.; Figurnov, M.; Ronneberger, O.; Tunyasuvunakool, K.; Bates, R.; Žídek, A.; Potapenko, A.; et al. 2021. Highly accurate protein structure prediction with alphafold. *Nature* 596(7873):583–589.

Jun, K.-S.; Li, L.; Ma, Y.; and Zhu, J. 2018. Adversarial attacks on stochastic bandits. *Advances in neural information processing systems* 31.

Li, D.; Zhao, D.; Zhang, Q.; and Chen, Y. 2019. Reinforcement learning and deep learning based lateral control for autonomous driving [application notes]. *IEEE Computational Intelligence Magazine* 14(2):83–98.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *Computer Science*.

O'Kelly, M.; Sinha, A.; Namkoong, H.; Tedrake, R.; and Duchi, J. C. 2018. Scalable end-to-end autonomous vehicle testing via rare-event simulation. *Advances in neural information processing systems* 31.

Qin, Z.; Tang, J.; and Ye, J. 2019. Deep reinforcement learning with applications in transportation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 3201–3202.

Senior, A. W.; Evans, R.; Jumper, J.; Kirkpatrick, J.; Sifre, L.; Green, T.; Qin, C.; Žídek, A.; Nelson, A. W.; Bridgland, A.; et al. 2020. Improved protein structure prediction using potentials from deep learning. *Nature* 577(7792):706–710.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. *nature* 529(7587):484–489.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *nature* 550(7676):354–359.

Tang, X.; Qin, Z.; Zhang, F.; Wang, Z.; Xu, Z.; Ma, Y.; Zhu, H.; and Ye, J. 2019. A deep value-network based approach for multi-driver order dispatching. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 1780–1790.

Tunyasuvunakool, K.; Adler, J.; Wu, Z.; Green, T.; Zielinski, M.; Žídek, A.; Bridgland, A.; Cowie, A.; Meyer, C.; Laydon, A.; et al. 2021. Highly accurate protein structure prediction for the human proteome. *Nature* 596(7873):590–596.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.;

Georgiev, P.; et al. 2019. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature* 575(7782):350–354.