Proceedings of the 3rd International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/4/2023322

Review on the application of deep learning algorithms in video game AI agent

Muqing Ge

School of Information Science and Technology, Northwest University, Xian, Shanxi, 710000, China

mg19179@essex.ac.uk

Abstract. AI algorithms have been applied in video games more than before. Since the early 2000s, the benchmark of the application of AI agent in video games has been put forward by researchers and engineers. This paper introduces the application of four different deep learning algorithms in video games, namely Component-based Hierarchical State Machine (CBHSM) algorithm, Monte Carlo Search Tree (MCTS) algorithm, Generative Adversarial Networks (GAN) algorithm, and the combination of A* algorithm and Q-learning algorithm. It can be summarized that both CBHSM and MCTS algorithms can modify and optimize the traditional Finite State Machine (FSM) algorithm applied in NPC (Non Player Characters) system. Additionally, A* algorithm can be combined with Q-learning algorithm to solve the problem of the difficulty on huge state space and limited time, and GAN algorithm, a model of unsupervised learning, can generate results by fewer training sets.

Keywords: A* algorithm, Q-learning, GAN, Finite State Machine, video game AI, Monte-Carlo Search Tree.

1. Introduction

So far, the NPC system of most games uses different AI algorithms to make the behavior patterns of characters more reasonable. Developers intend to use AI to improve the design of the game level and create the NPC with more authenticity. They also expect the enemies whom players must face to have more diverse response to the action of players. A simple NPC system can be easily penetrated by players and players will be bored soon. Moreover, the developer of the video game can test and improve the setting of the game level through AI agents. The test and development of AI agent on video game has become the benchmark to inspect the quality of AI algorithms. During game-playing, players need to respond to complex rules and contingencies, and they need the ability of dynamic-path planning, team-work, and learning. It is an attractive challenge for AI agent and many researchers aim to work on this field.

This paper first reviews the Component-based Hierarchical State Machine (CBHSM) algorithm. It is a type of modified Finite State Machine (FSM) algorithm. Then the Monte Carlo Tree Search (MCTS) algorithm applied in the NPC system is introduced. The next part describes the application of some algorithms in video games, including the combination of A* algorithm and Q-learning algorithm, as well as the Generative Adversarial Network (GAN) algorithm.

DOI: 10.54254/2755-2721/4/2023322

2. Application of AI algorithms in NPC system

The goal for the video game is to establish the NPC which is both interesting and changeable. In some video games, the NPC is designed to have changeless action models that can be easily recognized by players, thus making players get bored. While in other video games, the NPC is designed to be too powerful for players to complete, thus making them frustrated. In fact, the purpose of NPC is not to give the player an unbeatable opponent, but to maximize the player's engagement and experience over a long period of time. AI algorithms can achieve this goal. The NPC based on AI algorithms can be constantly improved as players explore the game contents so that players can keep themselves interested in the game.

2.1. Component-based hierarchical state machine

In fact, Finite State Machine (FSM) is a method quite popular for AI algorithm application in games. It is easy for developers to understand and operate the development process. The manageable method of operation also allows fast prototyping and enables more iterations for developing. Its lightweight makes it possible to be implemented in real-time systems. Compared to other models, FSMs are more suitable for validation and testing [4].

Traditional FSM requires developers to complete all the architecture for expected states and transitions during the development phase and the code is difficult to change after development. Therefore, FSM is prone to face combinatorial explosion problems (FSM states and transitions will be difficult to control when the environment complexity grows). In different environments or different games, developers have to recode the stations and transition before rather than using it repeatedly. This dilemma is solved by a modified FSM called CBHSM [4]. CBHSM is based on an asynchronous event-driven system (AEDS) which has the function of decoupling states and its context. In the CBHSM system, characters interact each other by the event exchange which means developers can change the state machine by adding or changing events instead of recoding the state machine. CBHSM has three advantages:

2.1.1. Compile time composability. In CBHSM design, hierarchy is the most important sign. When implementing a new state class, the developer should design a top-down structure and only consider the top design of class. The detailed states and behaviors are left to the subclass. Obviously, hierarchical design conforms to the thought of global design and significantly reduces the error in design. It also improves efficiency of developers.

2.1.2. Design time configurability. The configuration of traditional FSM of NPC design is usually hard-coded by programmers but game designers should be responsible for it. If the design changes, recompiling is necessary, which will waste too much time. The CBHSM puts off the work of hardcoding to the game design time. Thus, game programmers and designers can finish their work alone with high efficiency.

2.1.3. Run time flexibility. To make the action of NPC more unpredictable and resourceful, the state machine must have the ability to accept the parameters of the environment at running time. The component technology greatly improves the flexibility and adaptation of the state machine.

2.2. Monte carlo tree search

Monte Carlo Tree Search (MCTS) algorithm is one of the most efficient algorithms in AI agent of game. It is a kind of search technique that does not rely on professional field knowledge [5]. Therefore, developers do not need to learn a lot of domain-specific knowledge, which can save time for developing. In the NPC system based on traditional FSM, a robot gives a fixed method to respond to players' attack, then players will quickly learn how the robot moves and figure out the way to beat the robot. Compared with FSM, MCTS can make more flexible responses to players' action. The MCTS algorithm was created to solve the FSM repeatability problem. The way MCTS works is to first

DOI: 10.54254/2755-2721/4/2023322

identify all the operations available to the robot for the current situation. Then, for each possible action, how the player can respond will be analyzed. It then considers all possible actions for the player and what responses it can make, etc. There are mainly 4 stages in the application of MCTS, namely the selection, the expansion, the simulation, and the back propagation. At first, there is only one node on the search tree and it is needed to make a decision on this situation. Each node on the search tree contains three types of basic information: the situation represented, the number of visits, and the cumulative score. The steps of MCTS are shown in Figure 1.

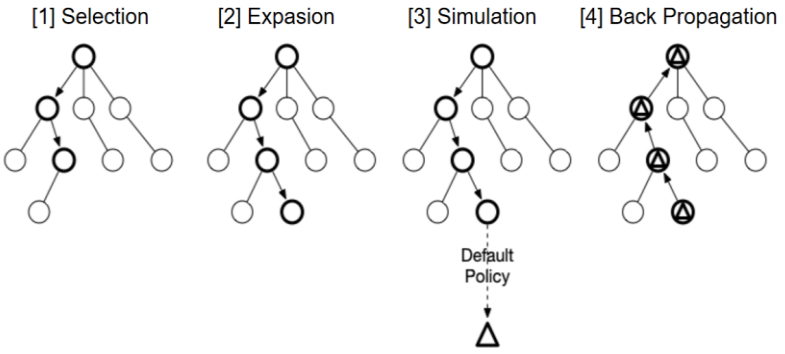

Figure 1. MCTS steps.

In some complex games, computer will make a huge number of decisions. It would take a long time for a computer to build a detailed tree for every possible choice and every possible scenario throughout the game. Therefore, to avoid the huge calculation, the MCTS algorithm randomly selects a few possible options and builds the tree only for the chosen one. That way, the calculations are faster and computer can analyze which option has the highest likelihood of a reward.

In a previous study, a MCTS algorithm is successfully applied in the game Dead-End. In this application, the longer the simulation time takes, the higher the chance of robot wins will be. The difficulty of the game can be decided by setting a time limit to plan the rate of the robot wins [6]. The study expounds that difficulty adjustment can be achieved by the time-limit of the MCTS algorithm, while the FSM algorithm can hardly achieve.

In another research, the authors discussed and applied a modified MCTS algorithm in general game playing. The modification of the algorithm is to create a parameter which can manipulate the probability when choosing a solution in evaluation function. With this function, the tree search will consider the most probably nodes at the beginning and give high evaluation to the nodes [7]. Every states will have a parameter P which is responsible for the probability of the outcome. This parameter will be used for evaluating the function.

In one research paper, the authors discussed and evaluated the eight enhancements for MCTS in general video game playing (GVGP): Progressive History, N-Gram Selection Technique, Tree Reuse, Breadth-First Tree Initialization, Loss Avoidance, Novelty-Based Pruning, Knowledge-Based Evaluations, and Deterministic Game Detection [5]. Some were proposed by predecessors, and some were the improvement made by the authors. The authors combined these enhancements to perform experiments and record data. The experiments show that most of the enhancements make improvements to the average winning percentage when being added individually into the game test. Specially, the Breadth-First Tree Initialization (BFTI) has a significantly improvement in the winning percentage when combined with most of the other enhancements such as the Tree Reuse, Knowledge-Based Evaluation, and Novelty-Based Pruning.

3. Application of AI algorithms in video games

The complex environment of video games gives AI algorithms a good application space. The special rule and mechanism of each video game requires developers to plan algorithms specifically for the game. That is an interesting challenge and many competitions of AI video games have been held. In

DOI: 10.54254/2755-2721/4/2023322

this part, the AI algorithm applied in the game Super Mario Bros in 2009 Mario AI Competition will be introduced. Some interesting and novel algorithms are also introduced in this section.

3.1. The combination of A algorithm and Q-learning algorithm*

In the Mario AI competition convened by IEEE society, the agent based on A* algorithm has a high mark [1]. In this section, a combined algorithm based on A* algorithm and Q-learning which has high efficiency will be introduced. This method solves the problem of huge state space and limited time. It also adds a learning ability to the search algorithm [2].

A* algorithm is an optimization algorithm which minimizes the cost of path. The cost f(n) is calculated by equation (1).

$$F(n) = g(n) + h(n) \tag{1}$$

f(n) is the estimation function of the movement cost from the initial state to the target state through the state n; g(n) is the actual cost of moving from the original state to the state n; h(n) is estimate of the movement cost from the state n to the target state; h(n) is calculated by heuristic function to estimate the distance to the goal node.

However, A* algorithm is not perfect to video games. Because of the limited search time, it is hard for agents to find a best path except a sub-optimal path. When agents face the dead ends, they could spin around and can not find their way out. This can be solved by adding the learning ability to agents. In other words, if agents can remember where they face a dead end, failures can be avoided. During the training phase, the agent remembers the location of branch points and effective actions. In the final experiment, the agent acts on the results of the A* algorithm except for the memorized location.

Q-learning is a type of reinforcement learning algorithms and is widely used for problem optimization. It is based on a decision rule set Q (st, at). st is the state of agent, at is the action that will be taken. Agents observe the location of Mario as a stat, and select a target node from A* algorithm as the action.

The combined algorithm has two stages: the searching stage and the learning stage. An agent will take the searching stage first by using A* algorithm. When the agent faces a dead end or can not find its way out, it takes into the learning stage. The agent will train a target node by the Q-learning method. When the agent detects the dead ends, it will get rewards. After detecting dead ends, the agent turns to the searching state.

In a previous study, the researcher made an experiment with the A* algorithm, Q-learning algorithm, and the combination of A* algorithm and Q-learning method. The level is set with a high wall, enemies, and dead ends. The combined algorithm got an excellent result while the other two single algorithms failed to pass the experiment [2].

3.2. Generative adversarial networks for game level generation

In traditional video games, there are only limited game levels for players to enjoy. For example, one of the most popular video games, the Super Mario Bros, only has 8 levels to play, and Contra only has 8 levels (rumors say that there are another eight hidden levels). Anyway, limited levels can not give players the best experience of game. However, the game level design is a complex component of game creation. Game designers must consider physics, difficulty, and playability in the creation process. Even a single level could take a lot of time to design.

GAN is composed of two important parts: Generator (G) and Discriminator (D). Data (mostly images) is generated by the generator in order to “treat” the discriminator. The aim of the discriminator is to find the “fake data” generated by the generator, thus determining if this image is real or generated by the machine. Thus, G and D constitute a dynamic “game process”. Firstly, fix the discriminator and train the generator until the data from the generator is strong enough to cheat the discriminator. Then, fix the generator and train the discriminator until the discriminator can distinguish true and false of the data from the generator. After several iterations, the desired training results are generated.

DOI: 10.54254/2755-2721/4/2023322

GAN have achieved impressive success in image generation. However, GAN faces challenges in the content generation of game levels, since it is difficult to combine the aesthetically appealing with playable levels at the same time. A kind of new GAN architecture called Conditional Embedding Self-Attention Generative Adversarial Network (CESAGAN) is purposed in a research. It is a modification of the self-attention GAN with an embedding feature vector input to condition the training of the discriminator and generator [3]. The research also purposed a new bootstrapping training processing. CESAGAN is used to combine functional requirements and alleviate the deficiency of training data. Advantages of CESAGAN include:

• It reduces the amount of information required to train discriminators. Most video games only have seldom levels which are available for the training set.

• It increases the quality. The levels generated by traditional GANs always have low quality and sometimes are unplayable.

• It increases the diversity. The diversity of levels has obviously been promoted compared to traditional GANs.

The first advantage is achieved by a self-attention mechanism. Because of the size limitation of the convolution kernel, the traditional convolutional GAN can only capture the relationships of local regions. In a self-attention GAN, the information of all positions can be utilized. Therefore, the system can use fewer data sets to obtain the same quality of training sets by the self-attention mechanism. It has an additional feature conditional vector to train the generator and discriminator.

Another method using fewer data sets for training is the SinGAN. It is the first unconditional generative model being trained based on a single natural image. In other words, the trained SinGAN can receive a random noise input to generate a new natural image [8]. This method fits AI video games well with very limited levels for training. Since the SinGAN is designed for RGB images, it can not generate the video game levels based on 2D token maps. In this section, a kind of GAN algorithm based on SinGAN called Token-based Oneshot Arbitrary Dimension Generative Adversarial Network (TOAD-GAN) will be introduced. The TOAD-GAN is successfully applied in the video game level generation. It is a new algorithm which can generate new levels of arbitrary size when being trained on a single training example [9].

The core part of this method, namely the SinGAN, is a new architecture which can create a generate model from a single image. Based on multiple GAN structures, SinGAN learns the distribution of image blocks with 11x11 resolution at different scales, and generates real images from rough to fine and low resolution to high resolution stage by stage. Therefore, SinGAN can isolate a large amount of information from an image for learning. In the TOAD-GAN, the SinGAN can be used to isolate information of one level and generate lots of levels based on the extracted information.

4. Conclusion

Among all the AI algorithms applied in video games, the A* algorithm and Monte-Carlo Tree Search algorithm are the most popular algorithms among video game developers in AI agent design. The combination of A* algorithm and Q-learning algorithm are mentioned in this paper. The agent can change the target by Q-learning algorithm to improve the efficiency. Monte-Carlo Search Tree is also a kind of improved algorithm which can replace the traditional FSM and select the node with the highest access frequency rather than selecting all the nodes. It improves the efficiency and has the ability of learning which allows the NPC to respond to the action of players. These two algorithms have simple structures that make computing processes faster. Another reason is that there is plenty of source code for both algorithms which saves developers lots of time. But we are still expected to see some important improvement of algorithm with pioneering significance. The Generation of game level is also an important part for game designers. Several improved GAN algorithms are mentioned in this paper. The adventure of the GAN algorithm is that it adapts to the situation of limited training sets, which meets the generation of the game level. In general video games, the method of dividing basic elements into several tokens rather than natural images can improve the efficiency of generation and make the generation available. The generation of game level based on the GAN algorithm inspires

DOI: 10.54254/2755-2721/4/2023322

game designers and save them a lot of time for design. One of the limitations of this paper is that the algorithms mentioned in this paper are limited because of the lack of data of some algorithms. The adventure of the similar improved algorithms do not have comparability because the variate of algorithms is not single.

References

[1] Togelius J, Karakovskiy S and Baumgarten R 2010 The 2009 mario ai competition. In IEEE Congress on Evolutionary Computation. number 1. 1–8.

[2] Shinohara S, Takano T, Takase H, Kawanaka H and Tsuruoka S 2012 Search algorithm with learning ability for mario ai – combination a* algorithm and qlearning. In 2012 13th ACIS International Conference on Software Engineering. Artificial Intelligence. Networking and Parallel/Distributed Computing. number 2. 341–344.

[3] Torrado R R, Khalifa A, Green M C, Justesen N, Risi S and Togelius J 2020 Bootstrapping conditional gans for video game level generation. In 2020 IEEE Conference on Games (CoG). 41–48.

[4] Hu W F, Zhang Q and Mao Y Q 2011 Component-based hierarchical state machine — a reusable and flexible game ai technology. In 2011 6th IEEE Joint International Information Technology and Artificial Intelligence Conference. volume 2. 319–324.

[5] Dennis J N J S, Chiara F S, Torsten S and Mark H M 2016 Winands. Enhancements for real-time monte-carlo tree search in general video game playing. In 2016 IEEE Conference on Computational Intelligence and Games (CIG). 1–8.

[6] Ma Y, He S J, Wang J P, Fu Y W and Shi Z Y 2010 Automatic AI design by the use of mcts for the game dead-end. In 2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery. volume 6. 2772–2776.

[7] Danil A C and Sergey A B 2020 Monte carlo tree search modification for computer games. In 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). 252–255.

[8] Shaham T R, Dekel T and Michaeli T 2019 Singan: Learning a generative model from a single natural image. CoRR. abs/1905.01164.

[9] Schubert F, Awiszus M and Rosenhahn B 2021 Toad-gan: a flexible framework for few-shot level generation in token-based games. IEEE Transactions on Games. 1–1.