Transactions on Computer Science and Intelligent Systems Research
ISSN: 2960-1800, eISSN: 2960-2238 | Volume 5
AIDML 2024

The Application of Artificial Intelligence in The Field Of Gaming

Yuhao Fu

College of Information Science and Technology Beijing University of Chemical Technology Beijing, China

2021040032@buct.edu.cn

Abstract. In recent years, the landscape of the electronic gaming industry has been profoundly reshaped by advancements in computer hardware and software capabilities. Notably, the proliferation of network and mobile gaming platforms has catalyzed unprecedented growth within the sector. This surge in popularity underscores the increasing demand for immersive and engaging gaming experiences across diverse demographics. This paper endeavors to delve into the intricacies of AI's role in gaming, providing an in-depth exploration of its core processes and application methodologies. Specifically, it aims to elucidate how AI techniques are leveraged to create intelligent characters and generate dynamic content, thereby enriching the gaming experience. By examining real-world examples of AI integration in gaming and forecasting future trends, this paper seeks to offer valuable insights into the evolving intersection of technology and entertainment.As technologies such as deep learning and neural networks continue to evolve, the application of AI in games will also become more intelligent and adaptive, able to adjust the gaming experience in real time based on player behavior and preferences.

Keywords: Artificial intelligence, game development, intelligent generation, deep learning.

1. Introduction

As computer technology advances, artificial intelligence has emerged as a key technology across various domains, including the gaming industry. The integration of artificial intelligence in games not only fosters innovation in game design but also enhances players' experience by offering more intelligent and challenging gameplay.With the rapid advancement of technologies like deep learning and reinforcement learning, the utilization of artificial intelligence in the gaming industry is poised to expand further [1]. Future games could leverage intelligently generated content to dynamically adapt game environments and narratives based on individual player preferences, thereby delivering more personalized and captivating gaming experiences. This article aims to explore the specific application techniques of artificial intelligence in games, focusing on its role in creating intelligent characters and generating intelligent content.

2. Game Method introduction

2.1. Reinforcement learning

Reinforcement learning is a machine learning paradigm that aims to achieve maximum cumulative reward on a task through the interaction of an Agent with the environment. In reinforcement learning, the agent tries different actions, observes feedback from the environment, and adjusts its behavior based on this feedback in order to maximize long-term rewards.

- The main elements:

Agent: The entity that makes decisions and takes actions.

Environment: The external system with which the agent interacts.

State: Information that describes the environment and, for the agent, is the basis for decision making.

Action: A specific action or decision taken by the agent.

Reward: Indicates the environment's feedback on the agent's behavior and is used to evaluate whether the behavior is good or bad.

- Basic process:

The agent observes the environment: At each time step, the agent observes the current state of the environment.

The agent selects action: Based on the observed state, the agent chooses to take a certain action.

Environment gives a reward: The agent's action causes a change in the state of the environment, and the environment rewards or punishes it accordingly [2].

Agent learning: The agent adjusts its behavior based on reward signals to gradually learn which actions to take in different states will maximize the cumulative reward.

Iteration: Repeat the above steps, iterating over time so that the agent gradually improves its strategy.

2.2. Deep Learning technology

Deep learning is an important branch of artificial intelligence, which can learn and understand complex patterns through neural network models. In games, deep learning is widely used in image recognition, speech synthesis, behavior prediction, etc., making the virtual world more realistic [3].

- Problem definition and data collection:

Clearly define the problem: Identify the problem to be solved and clearly define the task, such as image classification, object detection, speech recognition, etc.

Collect data: Collect data for training and evaluating the model, ensuring the quality and representativeness of the data set.

Data cleaning: Dealing with outliers, missing values, or errors in the data.

Data standardization: Standardizing data to ensure that different features have similar scales.

Data enhancement: Enhancements are made to the training data to expand the dataset and improve the generalization of the model.

- Model construction:

Choose the model architecture: Choose the appropriate deep learning model according to the nature of the problem, such as convolutional neural network (CNN), recurrent neural network (RNN) or Transformer.

Define the network architecture: Build the network structure of the model, including the hierarchy and the design of the neurons.

Select activation function and loss function: Select the appropriate activation function and loss function to ensure the learning effect of the model.

- Model training:

Forward propagation: The input data is passed through the model and the output of the model is calculated.

Backpropagation: Calculate the gradient using a backpropagation algorithm and update the model parameters to minimize losses.

- Model evaluation:

Validation set evaluation: Use validation sets to evaluate the performance of the model and adjust model hyperparameters to improve generalization.

Test set evaluation: Evaluate the final performance of the model using an independent test set, verifying its performance on previously unseen data.

- Model deployment:

Model export: Export the trained model into a deployable format.

Integration into an application: Integrate the model into a real application, be it an embedded system, a mobile application, or a cloud service.

- Model monitoring and maintenance:

Performance monitoring: Monitor the performance of the model in the production environment and check the performance of the model regularly.

Update and optimization: Update and optimize the model as needed to accommodate new data distributions or improve the performance of the model.

3. Applications

3.1. Intelligent Character

Reinforcement learning enables NPCS (non-player characters) to learn and adapt based on the consequences of their actions, through the agent's interaction with the environment, feedback on reward signals, and application of learning algorithms. Based on deep reinforcement learning, AlphaGo Zero developed by DeepMind completely abused and killed humans in the game of Go without using any human Go data; OpenAI's Dota Five reached the top level of human players in the game of Dota; DeepMind's AlphaStar also beat professional human players at StarCraft. These were milestone events in the development of deep reinforcement learning, demonstrating the power of deep reinforcement learning in these situations.

To apply reinforcement learning in any game or other scenario, the first thing to do is abstract the scenario into a Markov decision process. As shown in the figure 1, we take the Bot or NPC we want to train as an agent, and the game it is facing as the environment, from which the character obtains its current state and reward and acts accordingly. The state is used to represent the feature information of the current character, which can be directly the original information of the current game screen, and then encoded by technologies such as CNN to extract the features, or it can be semantic information extracted through the game API. Similarly, the actions of a character can be the same as the key information of a human player, or they can be directly controlled by a higher level of action API.

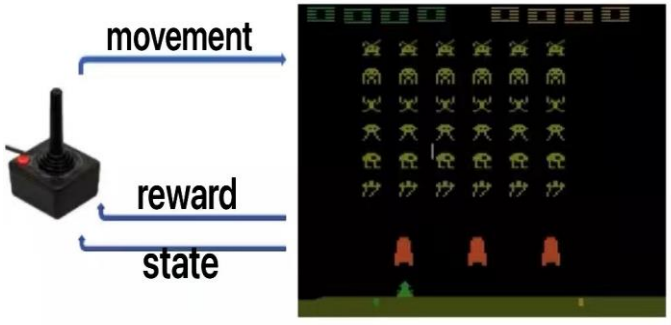

Fig. 1 Schematic diagram of the reward section (Photo/Picture credit: Original)

In the Figure 1, different rewards correspond to distinct learning objectives, which should be tailored based on the desired goals. In a basic game, a score accumulated over time or the outcome of winning or losing can serve as rewards [3,4]. However, in more intricate games, such simplistic reward setups may prove challenging to train or lead to slow convergence. This domain is commonly referred to as Reward Engineering in the realm of reinforcement learning applications.

Once the game is abstracted into an MDP, the appropriate reinforcement learning algorithm, such as DQN or PPO, can be chosen for training. Similar to other machine learning techniques, the hyperparameters, network architecture, and choice of learning algorithms in these methods will impact the learning outcomes. These factors should be fine-tuned based on the training progress, and the reward scheme should also be re-optimized accordingly [5].

3.2. The application of QQ Flying Car:

QQ Flying Car is a mobile racing game where players need to utilize a range of skills, including nitro acceleration, to enhance their driving speed and reach the finish line as quickly as possible. By leveraging deep reinforcement learning technology, we have developed a Bot AI for QQ Flying Car, which has already been deployed in a test environment. Let me provide a brief overview (Figure 2).

Fig. 2 QQ Flying Car game (Photo/Picture credit: Original)

To apply deep reinforcement learning, we first define the Markov Decision Process (MDP). In the state space, we extract various information such as speed, angle, distance to walls on both sides, distance to obstacles, etc., using the game's API and concatenate them into a vector to represent the current car state. For the action space, we utilize the game's API to map and combine controls like left turn, right turn, drift, small spray, big spray, and others to define our actions. In terms of rewards, a simplified approach is adopted: the reciprocal of the time taken to cover a distance is used as a positive reward to encourage faster driving, while collisions, going in the wrong direction, and other negative events are penalized with negative rewards. The reward function corresponding to a distance segment can be expressed by this formula:

$$r = 1/t - \alpha \cdot I_{collision} - \beta \cdot I_{wrong} \quad (1)$$

Where, α and β represent different weighting coefficients, respectively, indicating whether there is a collision and the wrong direction in a distance.

After formalizing the game scene into an MDP, we opted to employ the PPO algorithm for model training due to its fast convergence speed and effective performance. Following some parameter tuning, the trained model is now capable of autonomously driving the vehicle and has yielded promising outcomes [6,7]. The image displayed depicts the flying car AI trained through deep reinforcement learning.Certainly, the aforementioned represents just a rudimentary version, and there exist numerous other challenges to address. These include the need to enhance training speed to align with the game's new version launch timeline, accommodating multiple maps within the game, ensuring model generalization across various maps, mitigating peculiar model movements, and establishing diverse skill levels for the model – from novice to expert – to enable its applicability in diverse scenarios. Addressing these issues will necessitate further efforts in algorithm design and reward engineering.

3.3. Content generation

Procedural Content Generation (PCG) is a field within game AI that refers to methods used to generate game content autonomously or with limited human input. PCG technology allows for the creation of in-game levels, map terrains, in-game multimedia content, and more, aiming to decrease the burden on game developers and artists, enhance efficiency, and cut down on game production costs [8]. Notably, algorithms have been utilized for map generation since as early as the 1973 game Maze War. Presently, advanced methods like neural networks and adversarial generative networks are being incorporated into PCG to advance game productivity. For example, Nvidia has developed a technology that can actively generate textures and materials in games based on photos, as well as track and render light in real time; A number of researchers are experimenting with automated level generation in games like Super Mario and Angry Birds; For example, in the new game end game launched by Tencent Game Happy Landlord in 2018, a large number of game end games are automatically generated through deep learning technology, as shown in the figure 3 is one of the levels.

Fig. 3 The game of Happy Landlord (Photo/Picture credit: Original)

3.4. Generate character animation

In modern games, creating lifelike characters with smooth and organic movement is crucial, known as character animation. Over time, various solutions have been developed, starting from Sprite animation and rigid hierarchy animation to today's advanced skin animation, resulting in improved effects. Nevertheless, accurately simulating human and animal movement remains a complex challenge, with limited methods available to replicate the diverse behaviors seen in the real world. Manual design of control logic is labor-intensive and struggles to adapt to new scenarios and situations efficiently.

Reinforcement learning provides a possible idea for motor synthesis, that is, the agent learns by trial and error during repeated actions to perform various skills in order to reduce the heavy dependence on human power. However, simply allowing the agent to explore freely may lead to some meaningless actions that do not affect the goal of the skill, such as irrelevant upper body movements, awkward postures, etc. The movements can be made more natural by incorporating more realistic bioengineering models. But building high-fidelity models is very difficult, and the resulting movements may still be unnatural. Therefore, an optimal system for automatically generating character animations should start by providing the agent with a predefined set of reference actions that satisfy the criteria. Subsequently, it should generate actions that align with the objectives and are physically plausible based on this foundation [9]. For instance, DeepLoco employs deep reinforcement learning to replicate motion data, while Generative Adversarial Imitation Learning (GAIL) leverages generative adversarial imitation learning to produce actions, both yielding promising outcomes.The Berkeley researchers went on to create DeepMimic, which mimics motion better. Once again, states and motion Spaces need to be defined according to the scene to be trained, but the reward section is slightly different, introducing reference clips as the basis for the reward. Specifically, It is essential to provide the agent with reference motion segments, such as a roundhouse kick or a backflip, for instance. These segments can be derived from motion capture data of humans or hand-drawn animations [10]. Subsequently, the trajectory error between the agent's actions and the poses of these specified reference actions can be

translated into rewards. Deep reinforcement learning can then be employed to train the agent to generate coherent motion sequences. Moreover, to enhance training efficiency and effectiveness, DeepMimic has also introduced two additional methods, which is referred to as reference state initialization, where, to enhance sampling efficiency, the agent is initially set to a randomly sampled state from the reference action at the start of each trajectory. The second technique, known as early termination, aims to boost simulation efficiency by prematurely ending trajectories that are deemed unlikely to yield satisfactory results, rather than prolonging the simulation unnecessarily. Following these procedures, the agent can acquire the ability to perform a diverse range of challenging maneuvers, as illustrated in the figure 4.

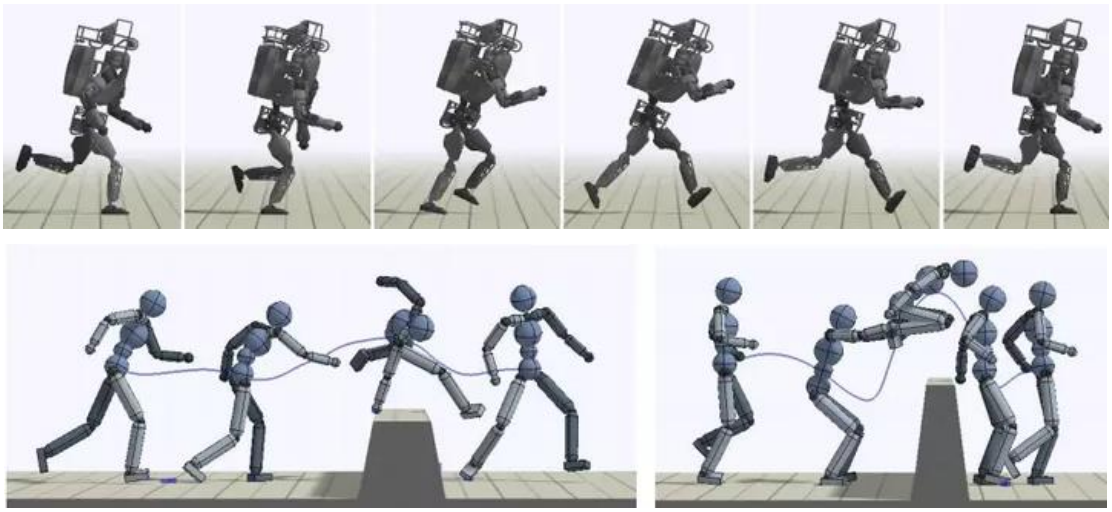

Fig. 4 The robot learns difficult tricks [10]

4. facing problems and Developing trend

4.1. Facing problems

Computing resource limitations: Some complex AI algorithms require a lot of computing resources, and gaming platforms may not be able to provide enough computing power.

Game content generation: How to use artificial intelligence technology to generate more rich and diverse game content to improve the play ability and fun of the game.

Game balance: In multiplayer online games, how to use artificial intelligence technology to achieve game balance and ensure the fairness and fun of the game.

Game emotion recognition: how to give AI systems a better understanding of players' emotional states to provide a more personalized game experience.

Game security: How to use artificial intelligence technology to detect and prevent cheating in games to ensure the fairness and security of games.

4.2. Development trends

Augmented reality and virtual reality: Artificial intelligence technology will be combined with augmented reality (AR) and virtual reality (VR) technology to provide a more immersive gaming experience for players.

Personalised gaming experience: By analysing the behaviour and preferences of players, AI systems can provide players with a personalised gaming experience, thereby improving the appeal and playability of games.

Adaptive game difficulty: The AI system can adjust the difficulty of the game according to the skill level and performance of the player to make the game more challenging and interesting.

Generative game design: AI technology can be used to generate game content, including maps, missions, characters, etc., to reduce the cost and time of game development.

Multi-agent systems: AI systems can be used to design and optimize multi-agent systems so that characters and NPCS in a game can act and interact more intelligently and realistically.

5. Conclusion

This paper first introduces the process of reinforcement learning and deep learning to help readers better understand how reinforcement learning, and deep learning technology apply to the field of games, and then introduces some applications of artificial intelligence technology in the field of games and introduces in detail the reward engineering technology in deep learning and the automatic generation technology in deep learning generated by the game "fight landlord" endgame mode. In general, players have higher and higher requirements for game artificial intelligence, and each game manufacturer pays more and more attention to this aspect. In particular, the game artificial intelligence design method based on machine learning will be widely used in the future.

References

[1] S. Tang, M. Hanneghan. State-of-the-Art Model Driven Game Development: A Survey of Technological Solutions for Game-Based Learning.Journal of Interactive Learning Research, 2011, 22(4):551-605.

[2] K. Xiang, R. W, Neurosurgery D O. Artificial Intelligence and Its Application in Medical Field.Journal of Medical Informatics, 2016., 15(6): 65-68.

[3] X. Chen, S. Xu, Y. Cao, et al.Application of artificial intelligence based on machine learning.Journal of Physics: Conference Series, 2020, 1693(1):012084 .

[4] C. Tang, Z. Wang Z, Sima X, et al. Research on Artificial Intelligence Algorithm and Its Application in Games, International Conference on Artificial Intelligence and Advanced Manufacture. 2020.

[5] F. Zhou. Application Status and Prospect of Artificial Intelligence in Game Development, Journal of Electronic Science and Technology, 2022, 49(2): 228-234.

[6] J. Hu. Research and Application of Artificial Intelligence in Game Development, Computer Science and Application, 2023, 45(8): 112-116.

[7] W. Bu Research on Intelligent Technology Based on Rules and Machine Learning in Games, Journal of Intelligent Systems, 2024, 34(6): 789-794.

[8] J. Fang. Game Artificial Intelligence in Computer Games, Computer Science, 2022, 52(4): 560-565.

[9] B. Hu. Research and Implementation of Artificial Intelligence-assisted Electronic Games, Software Engineering, 2023, 29(3): 398-402.

[10] C.Y. Yang. Behavior Decision of Game Agents Based on Artificial Intelligence, Pattern Recognition and Artificial Intelligence, 2023, 24(2): 180-185.