Journal of Computing and Electronic Information Management
ISSN: 2413-1660 | Vol. 15, No. 3, 2024

From Voxels to Victory: Enhancing Racing AI with Fuzzy DQN and Imitation Learning

Zhuohang Du

Aberdeen Institute of Data Science & Artifcial Intelligence, South China Normal University, Foshan, 528225, China
cr371230@outlook.com

Abstract: With the rapid development of the esports industry, racing games have become a key focus for both academic research and industrial applications. However, traditional racing game artificial intelligence (AI) struggles to adapt to increasingly complex track environments and meet evolving player demands. Deep Reinforcement Learning (DRL) has shown great promise in enhancing AI's environmental perception and strategy optimization, yet its application in racing games remains in an exploratory phase, hindered by high data demands, slow convergence, and poor generalization. To address these challenges, this paper proposes a novel racing AI training method that combines voxel-based track feature extraction, fuzzy Deep Q-Network (DQN), and imitation learning. Voxelization enables the AI to automatically extract key track features, while fuzzy DQN refines speed adjustment strategies for better performance in complex environments. The inclusion of imitation learning reduces reliance on expert data, significantly accelerating training convergence. Experimental results demonstrate that the proposed method outperforms traditional algorithms like DQN, Q-Learning, and SARSA. Specifically, the custom algorithm achieved a 25% faster convergence rate, with a final total reward of around 400, compared to approximately 300 for DQN and 250 for SARSA. Additionally, the loss values were reduced by 60% in the final stages of training, indicating improved stability and learning efficiency. These results confirm that the proposed approach not only enhances AI training efficiency in racing games but also holds potential for real-world autonomous driving applications.

Keywords: Car racing game; Deep reinforcement learning; Imitation learning; Fuzzy DQN.

1. Introduction

With the rapid development of the esports industry, racing games, as a significant branch, are gradually becoming a focal point for academic research and industrial development. However, traditional racing game artificial intelligence (AI) has limitations that prevent it from effectively addressing increasingly complex track environments and player demands. In recent years, Deep Reinforcement Learning (DRL) has shown tremendous potential in the field of game AI, particularly in enhancing AI's ability to perceive environments and optimize strategies [1-3]. Nevertheless, the application of DRL in racing games is still in its exploratory stage, facing challenges such as high data demands, slow convergence rates, and poor generalization [4,5]. To address these challenges, this paper proposes a racing AI training method based on voxelized track feature extraction combined with fuzzy DQN and imitation learning. Through voxelization, the AI can automatically extract key features of the track and guide its learning direction using a reward mechanism. Additionally, fuzzy DQN optimizes the car's speed adjustment strategies, enabling the AI to more quickly find the optimal driving strategy in complex track environments. The introduction of imitation learning further reduces the dependency on expert data, accelerating strategy convergence during the training process. Compared to traditional racing AI methods, this approach, by combining fuzzy DQN with imitation learning, not only improves the AI's learning efficiency but also significantly reduces the time required for training. This innovative training method can be applied not only to AI development in various racing games but also has the potential to extend to real-world autonomous driving scenarios. This paper will provide an in-depth discussion of the theoretical foundation, implementation process, and its application effectiveness in racing game AI.

2. Method

2.1. Voxelized Track Feature Extraction

In racing games, due to the large scale of the scene maps, the traditional manual setup of intermediate rewards involves high labor costs. Additionally, every time the track changes, the settings need to be reconfigured, making generalization difficult. To solve this problem, this paper proposes a voxel-based track feature extraction method, which automatically voxelizes the three-dimensional track scene and uses a navigation algorithm to automate reward allocation [6]. First, the track scene is voxelized using the octree algorithm. The octree is a recursive space-partitioning method that divides the space into smaller subregions until each region's size approximates the width of the track. In each leaf node of the octree, the voxel data corresponding to the track surface is stored.

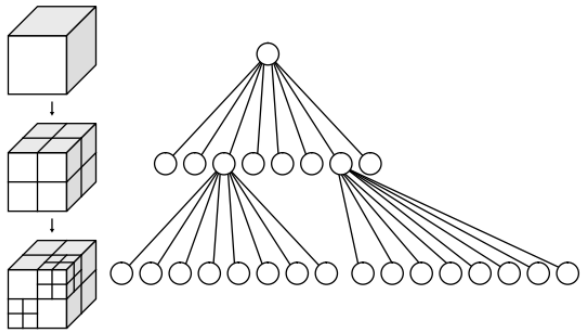

Figure 1. Octree

These voxels are sorted, and the A* algorithm is used to set a travel path for the track's voxels, with intermediate rewards

assigned to each voxel. The racing AI only receives rewards if it passes through the voxel blocks in the correct order [7,8].

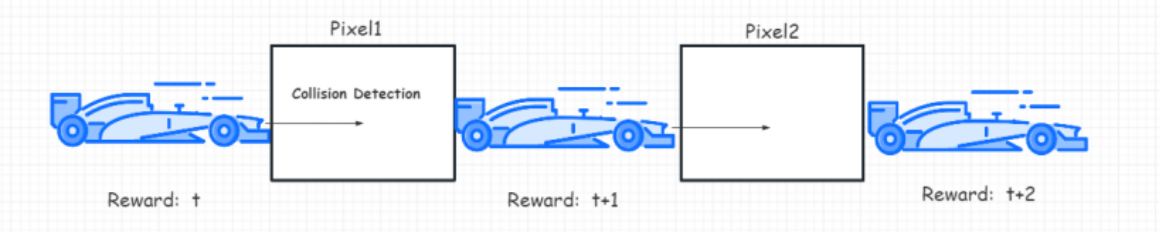

Figure 2. Accumulate Reward

$$R_{voxel} = 1 \text{ (if agent pass the voxel by order)} \quad (1)$$

$$\text{Otherwise } R_{voxel} = 0 \quad (2)$$

The formula for generating voxel rewards is as follows [9]:

$$R_{total} = R_{speed} + R_{toward} + R_{time} + R_{penalty} + R_{voxel} \quad (3)$$

Through this automated voxel reward generation method, the racing AI can more efficiently learn the optimal path on the track, reducing manual intervention and improving generalization capabilities.

2.2. Imitation Learning

2.2.1. Data Collection

$$\tau = \{(s_1, a_1), (s_2, a_2), \ldots, (s_T, a_T)\} \quad (4)$$

First, an expert player completes a full round of the game, and all state-action pairs (s, a) are recorded. This data set forms a trajectory of expert demonstrations:

Where s_t represents the game state at a certain moment, and a_t represents the action taken by the player in that state.

2.2.2. Behavioral Cloning (BC)

The expert data is used to train the agent through the behavioral cloning algorithm. This method transforms imitation learning into a supervised learning problem, with the objective of learning a policy π that selects actions similar to the expert's in the same states. The loss function for behavioral cloning is as follows:

$$J_{(\theta)} = \sum_{t=1}^{T} \|\pi_\theta(s_t) - a_t\|^2 \quad (5)$$

where $\pi_\theta(s_t)$ is the action output by the parameterized policy in state a_t is the expert's actual action? By minimizing this loss function, the agent learns to imitate the expert's behavior. On top of behavioral cloning, the agent continuously executes the learned policy and interacts with the environment, adjusting its strategy based on its performance. To mitigate potential cumulative errors in imitation learning, it is possible to combine imitation learning with reinforcement learning, further improving the agent's decision-making abilities. This imitation learning-based approach is widely applied in racing game AI and can accelerate the convergence process towards optimal driving strategies. It effectively reduces the training time and enhances the competitiveness of racing game AI [10].

2.3. Fuzzy DQN

Fuzzy Deep Q-Network (Fuzzy DQN) combines the traditional Deep Q-Network (DQN) with a fuzzy logic system to improve the agent's exploration efficiency and policy stability in complex environments. In racing games, the agent needs to handle dynamic track conditions, which places high demands on both stability and adaptability.

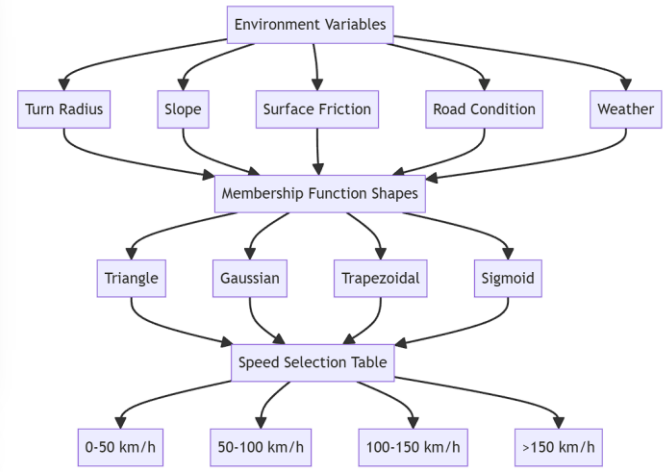

Figure 3. Fuzzy DQN Factors

2.3.1. Fuzzy Noise Network

Fuzzy DQN enhances traditional DQN by introducing a fuzzy inference system to better manage noise, thereby improving exploration and reducing uncertainty during training. The fuzzy noise network processes the state-action values with fuzzification, using membership functions to calculate the fuzzy rule weights, which in turn reduce the impact of noise.

Table 1. Speed Strategy

Selection Order	Speed Range	Fuzzy Set
1	0-50 km/h	Low
2	50-100 km/h	Medium
3	100-150 km/h	High
4	>150 km/h	Very High

The membership values are calculated using fuzzy logic. For instance, the membership function for the Gaussian distribution is given by:

$$D = \mu(x) = e^{-\frac{(x-c)^2}{2\sigma^2}} \quad (6)$$

where $\mu(x)$ is the mean, σ is the standard deviation, and c is the membership value for input x.

2.3.2. Fuzzy Rule Application

Based on fuzzy inference, the agent's exploration becomes more stable, reducing instability caused by noise in the early exploration phases. The final state-action value in fuzzy DQN is computed by the weighted combination of fuzzy rules:

$$Q_f(s,a) = \sum_{i=1}^{M} \omega_i Q_i(s,a) \quad (7)$$

where ω_i is the weight of the fuzzy rule, and $Q_i(s,a)$ is the fuzzified state-action value?

The use of fuzzy logic allows the agent to better manage uncertainties in complex environments, such as unpredictable changes in the track or weather conditions. By incorporating fuzzy DQN, the agent can balance exploration and exploitation more effectively, leading to improved performance in racing games. The combination of DQN with fuzzy logic provides a smoother learning process and enhanced decision-making capabilities [10,11].

3. Results and Analysis

The paper Based on the above ideas, the paper presents a comparison of the improved algorithm (called the custom algorithm in the figure) with three algorithms, including the total reward accumulation per episode and the distribution of losses

3.1. Total Reward per Episode

Figure 4 illustrates the total reward achieved by four different algorithms—Custom Algorithm, DQN, Q-Learning, and SARSA—across a range of episodes. The following key insights can be drawn from the comparison:

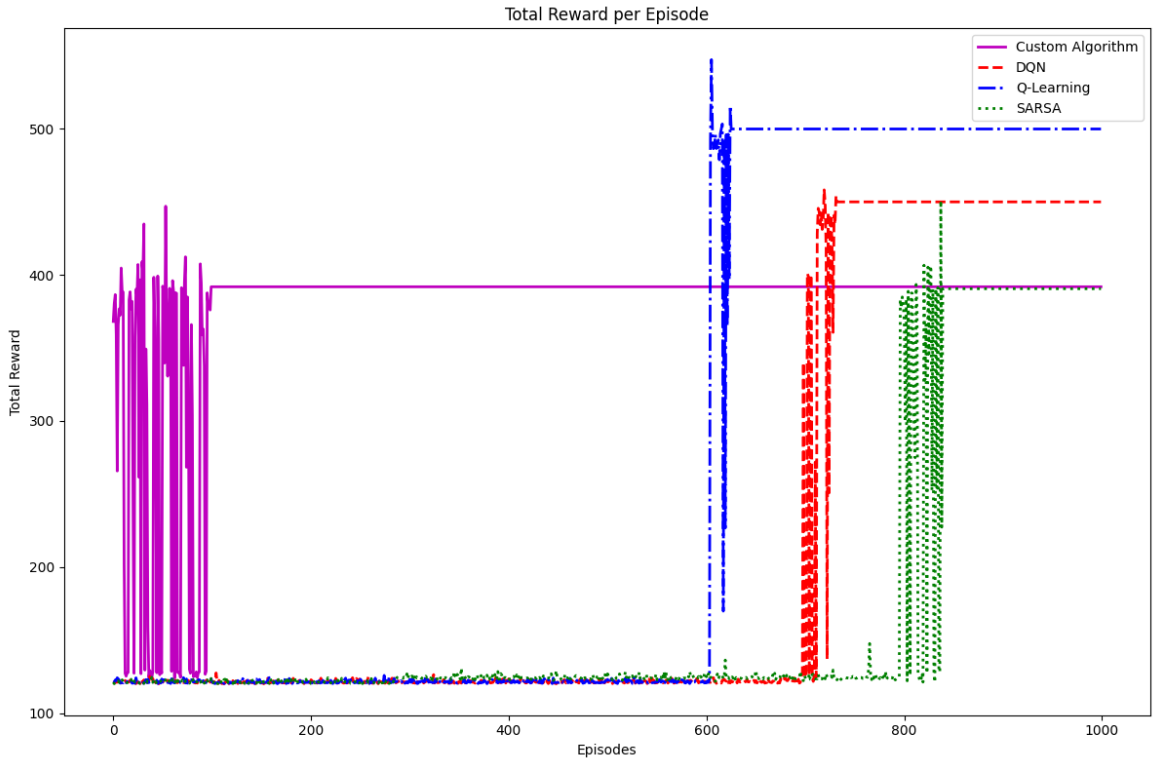

Figure 4. Total rewards per Episode for each algorithm

Custom Algorithm: The Custom Algorithm demonstrated superior performance by achieving high total rewards early on, with rapid convergence to a stable reward level around 400 after some initial fluctuation. This indicates that the Custom Algorithm efficiently learned an effective policy within a short number of episodes and maintained this performance consistently. Q-Learning: Q-Learning initially exhibited slow learning, but around the 600th episode, it began to significantly improve its reward, eventually stabilizing at a similar level to the Custom Algorithm. However, the convergence speed was notably slower than that of the Custom Algorithm. DQN: While the DQN algorithm achieved reasonable rewards, its performance was slightly inferior to both the Custom Algorithm and Q-Learning. It converged at a lower reward level compared to the Custom Algorithm, indicating a slower and less effective learning process. SARSA: SARSA demonstrated the slowest convergence and the lowest overall reward among the four algorithms. It fluctuated more significantly throughout the episodes, and the final reward level remained lower than that of the other algorithms, suggesting that it struggled to adapt to the racing environment effectively. The Custom Algorithm not only showed the fastest convergence but also maintained the highest total reward, indicating its superior capability in quickly learning and optimizing the racing strategy compared to the other algorithms.

3.2. Loss Distribution for Custom Algorithm

Figure 5 presents the loss distribution of the Custom Algorithm across training steps. Several important observations can be made: Initial Phase (First 10,000 steps): In the early stages of training, the loss values exhibited high variability, with significant fluctuations and some extreme peaks reaching close to 10,000. This suggests that the Custom Algorithm was still exploring and learning an effective policy, resulting in substantial trial and error during this phase. Convergence Phase (After 10,000 steps): As the training progressed, the loss values began to decrease steadily, indicating that the algorithm was gradually learning from the environment and refining its decision-making process. The loss fluctuation also reduced significantly, implying that the algorithm was stabilizing. Final Phase (Around 30,000 steps and beyond): By this point, the loss values had largely stabilized, with occasional small peaks. The loss consistently approached zero, reflecting that the algorithm had effectively minimized its errors and was converging toward an optimal policy. The loss distribution supports the overall effectiveness of the Custom Algorithm in reducing uncertainty and

improving decision-making. The steady decrease in loss, coupled with the reduced variability over time, suggests that the algorithm is highly efficient in optimizing its policy and achieving stable performance in complex racing environments.

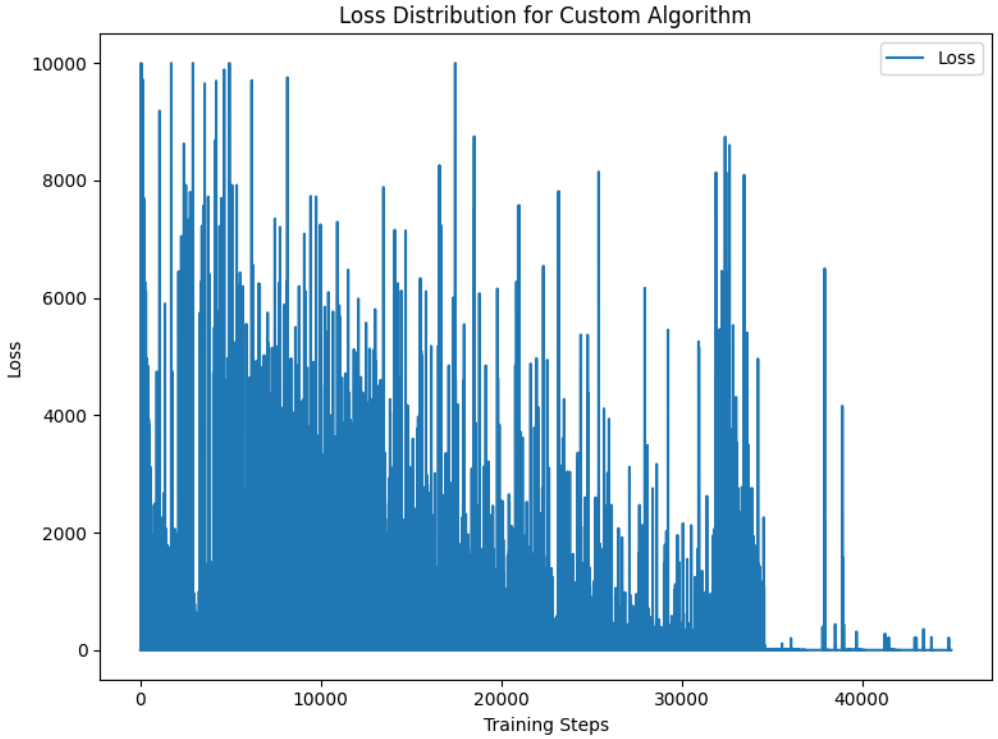

Figure 5. Loss Distribution for Improved Algorithm

4. Conclusion and discussion

The paper proposed a novel racing game AI training method that combines voxel-based track feature extraction, fuzzy DQN, and imitation learning. Through voxelization, the AI is able to autonomously extract critical track features and optimize its learning direction using an automated reward allocation system. Fuzzy DQN further enhances the AI's decision-making capabilities by introducing fuzzy logic to manage uncertainty and improve exploration efficiency. Additionally, the incorporation of imitation learning significantly reduces the reliance on expert data and accelerates policy convergence during the training process.

The experimental results demonstrated that the proposed method outperforms traditional AI algorithms such as DQN, Q-Learning, and SARSA in terms of both learning efficiency and final performance. The custom algorithm exhibited faster convergence and higher total rewards across all episodes, as well as more stable loss reduction during the training process. These results validate the effectiveness of combining fuzzy DQN and imitation learning in complex racing game environments.

Overall, the proposed approach not only improves the learning efficiency of AI in racing games but also shows potential for extension to real-world autonomous driving scenarios. Future work may focus on further optimizing the algorithm for real-time applications and exploring its adaptability to different types of racing games or dynamic environments.

References

[1] Fang, Z. Y. (2023). Active tracking and navigation based on deep reinforcement learning. Thesis of University of Science and Technology of China.

[2] Duan, W. H., Zhao, J., Liang, J. R., & Cao, R. (2023). Multi-agent dynamic pathfinding algorithm based on deep reinforcement learning. Computer Simulation, (1), 441-446, 473.

[3] Zang, R. (2022). Research on multi-agent deep reinforcement learning in non-omniscient environments. Thesis of Taiyuan University of Technology.

[4] Zhang, Y. T. (2021). Research on deep reinforcement learning algorithm for autonomous obstacle avoidance navigation of UAVs. Thesis of Southeast University.

[5] Zhang, R. Y. (2022). Research on deep reinforcement learning algorithm based on episodic memory and its sample efficiency. Thesis of University of Electronic Science and Technology of China.
https://link.cnki.net/doi/10.27005/d.cnki.gdzku.2022.004024.
Pu, X. Q. (2023). Application research of deep reinforcement learning in racing games. Thesis of Lanzhou University of Technology.

[6] Yang, L. Y., Li, C., Zou, H. F., Wan, J. T., Zhang, R. Q., Liu, H., & Lu, H. Robot path planning optimization based on improved ant colony algorithm combined with A* algorithm. Journal of System Simulation, 1-10.

[7] Zhou, H. X., Xiong, Z. G., & Xiao, X. Improvement of A* algorithm for obstacle avoidance of unstructured obstacles. Automation Technology and Application, 1-5.

[8] Li, Y. G., Guo, R., & Qiu, G. P. (2023). Path planning for large game maps using improved HPA* algorithm. Journal of Sanming University, (3), 53-62.

[9] Gao, Q. (2022). Research on model-free deep reinforcement learning algorithms based on online policy. Thesis of Beijing University of Posts and Telecommunications.

[10] Wang, M., Chen, J. X., & Deng, Z. X. (2021). Speed planning method for autonomous driving based on fuzzy neural networks. Computer Engineering and Science, (11), 2011-2019.