Research on Robot Path Planning Method Based on Deep Learning Q Algorithm

Haochen Lu

[1]School of Integrated Circuits, Anhui University, Anhui, China
*WB2224170@stu.ahu.edu.cn

Abstract. With the rapid development of mobile robot technology, dynamic decision-making and path planning of robots in complex dynamic environments have become a current research hotspot. Due to its advantages in handling dynamic environments, the Q-learning algorithm in deep reinforcement learning has been widely applied to robot path planning tasks. This paper reviews the current applications of deep Q-learning algorithms in path planning. Then it analyzes the existing challenges, and proposes directions for further improvement. This paper concludes that in practical applications, deep Q-learning algorithms still face several challenges, such as the curse of dimensionality, low data efficiency, long decision delays, and a tendency to fall into local optima. To address these problems, researchers have proposed some improvement measures, including prioritizing experience replay, adding a noise layer, introducing causal models, and other strategies. In the future, when studying path planning in dynamic environments, the research findings of this paper provide valuable directions for further investigation.

Keywords: Deep Learning, Path Planning, Q-Learning Algorithm, Prioritized Experience Replay, Causal Model

1 Introduction

As industrial manufacturing moves towards an era of intelligence and automation, robots are widely used in fields such as industry, agriculture, healthcare, and transportation. However, as the working scenes of robots are no longer limited to static environments, how mobile robots can autonomously navigate and perform tasks in complex dynamic environments has become a research hotspot in the field of artificial intelligence. In order to enable intelligent robots to have the ability to autonomously learn and perform tasks in complex environments and improve their local path planning capabilities, the innovation of traditional path planning algorithms has become a hot research area. The traditional methods based on heuristic and geometric algorithms (such as the A* algorithm and Dijkstra ' s algorithm[1]. Although it performs excellently in static environments due to its simplicity and reliability, in dynamic, unstructured environments, it suffers from high computational complexity and insufficient adaptability, making it difficult to effectively handle environmental uncertainties[2]. The currently popular path planning algorithms have better

J. Iqbal (ed.), *Proceedings of the 2025 2nd International Conference on Mechanics, Electronics Engineering and Automation (ICMEEA 2025)*, Advances in Engineering Research 272,
https://doi.org/10.2991/978-94-6463-821-9_27

convergence characteristics and improved decision-making accuracy by replacing experience, introducing prior knowledge, and optimizing the state-action space.

In recent years, in the research on robot path planning algorithms at home and abroad, deep learning (DL), as an important branch of artificial intelligence, provides a new solution to the path planning problem by combining the feature extraction ability of deep learning and the decision-making ability of reinforcement learning. The Q-learning algorithm is widely used in robot path planning tasks [3]. The Q-learning algorithm [4]was first proposed by Watkins in 1989. After years of development, it has been widely applied in areas such as agent control, path planning, and finance. Currently, the performance of the Q-learning algorithm is mainly improved through methods such as experience replay, double Q-learning[5]and adjusting the greedy strategy.

With the introduction of Deep Neural Networks (DNN), the Q-learning algorithm has further evolved into the Deep Q Network (DQN) [6], which can effectively handle high-dimensional state spaces and has improved data utilization efficiency. The use of experience replay[7] and the reference to the objective function have significantly improved the model's training stability and data efficiency, among other advantages, making DQN a promising research frontier.

This article will explain the current research status at home and abroad, the basic principles of DQN, the shortcomings of existing algorithms, and improvement measures. The aim of this study is to address the issues of unstable decision-making and the tendency to fall into local optima when mobile robots perform autonomous navigation in complex dynamic environments. By combining deep Q-networks with prioritized experience replay, double Q-learning, dynamic noise, and competitive frameworks, this method significantly improves the convergence speed and decision accuracy of the path planning algorithm. Ultimately, this research provides new insights for achieving efficient path planning for robots in dynamic, unstructured environments and contributes to the development of intelligent robotics technology

2 Basic Principles of DQN

DQN is a classic algorithm in the field of Reinforcement Learning. It combines the advantages of traditional Q-learning and deep neural networks, significantly improving the performance of reinforcement learning in high-dimensional state spaces. The original idea was proposed by Google’s DeepMind team in 2015 [8], The principle is to use deep learning networks to approximate the Q-values, enabling the agent to learn the optimal solution in complex, high-dimensional environments. Compared to traditional Q-learning algorithms, DQN can better approximate complex functions, significantly improving the ability of intelligent robots to autonomously perform tasks in complex environments. The specific principle of DQN is to use a deep neural network to approximate the function that evaluates the value of different actions for each state in traditional Q-learning. After the agent takes an action in a certain state, it receives an immediate reward and transitions to the next state. The network gradually adjusts its parameters based on the difference between the estimated current value, the actual received reward, and the highest possible future reward. In this way, the neural

network can continuously learn and optimize in complex, high-dimensional state spaces, helping the agent choose the optimal action, thereby achieving smarter autonomous decision-making and task execution. The core of the DQN algorithm is based on the Q-learning algorithm, combined with deep learning. It uses the state as input to the neural network, allowing the network to output a set of actions for the current state, from which the optimal action is then chosen for execution. This algorithm plays a pioneering role in solving the curse of dimensionality and improving data utilization efficiency.

2.1 Combining Convolutional Neural Networks with Q-Learning Algorithms

DQN uses convolutional neural networks as its framework and replaces the value function with network approximation.

$$f(s, a, w) \approx Q^*(s, a) \qquad (1)$$

$\mathbf{f(s, a, w)}$ is the value function, which is ultimately replaced by the Q-values output by the neural network. The schematic diagram is shown in Figure 1：

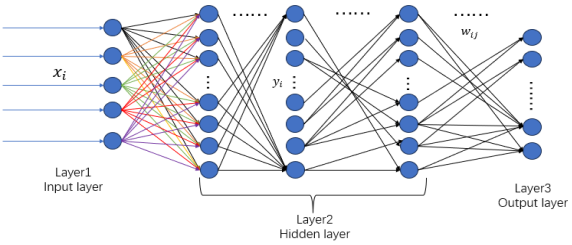

Fig. 1. Schematic diagram of convolutional neural network (Picture credit : Original)

As shown in Figure 1, deep neural networks are layer-by-layer connected, where the output of the previous layer serves as the input to the next layer. Sometimes, intermediate noise layers are added to increase the fitting accuracy of the samples. Assuming that layer l-1 has m neurons, the i-th neuron of layer l can be expressed as:

$$y_i^l = f\left(\sum_{j=1}^{I} w_{ij}^l \, a_j^{l-1} + b_i^l\right) \qquad (2)$$

This is the recursive formula for a neural network.

2.2 DQN uses the experience replay mechanism to store sequence samples for training

Figure 2 shows the process of the DQN algorithm. In DQN, two independent networks are used to process data: the target network and the current value network. Before data processing, the environment variables are passed to the current value network. After taking random actions, the new state and reward values are obtained and stored in the buffer. Data is then retrieved from the buffer and used to train the target network.

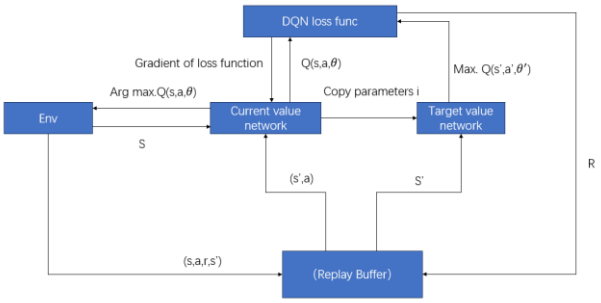

Fig. 2. DQN Algorithm Flow (Picture credit : Original)

Doing so can reduce many unnecessary comparison processes and reduce the frequency of exploration in the later stages of the algorithm implementation, thereby achieving rapid convergence while maintaining stability.

2.3 2.3DQN uses memory playback to save and record data

In the process of convolutional operations with the environment, there is a high correlation between the previous and subsequent states, which can lead to overfitting. Experience replay refers to the random sampling of samples from the memory buffer during each learning process, and learning is conducted with gradients. By using experience units from the memory matrix, new experience units can be mixed with previously trained ones, thereby breaking the correlation between adjacent samples and improving sample utilization.

The specific sample playback process is shown in Figure 3:

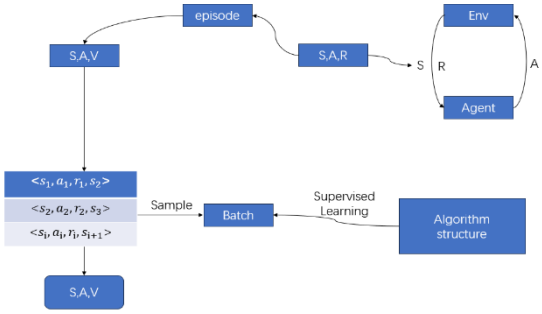

Fig. 3. Sample playback process (Picture credit : Original)

3 Research and application status at home and abroad

3.1 DQN uses memory playback to save and record data

The Q-learning algorithm suffers from the curse of dimensionality when handling high-dimensional problems, which exacerbates the data sparsity issue and further reduces learning efficiency. To address this problem, Google's DeepMind team [8] proposed integrating deep neural networks with experience replay strategies into the learning process. This not only avoids the shortcomings of Q-learning when dealing with high-dimensional problems but also optimizes fitting accuracy under nonlinear conditions. The main improvement step is to set up an experience replay unit, which aims to reduce the number of training samples required. The core of the experience replay mechanism lies in assessing the "value" of samples. During training, a prioritized sampling strategy is used to select samples from the experience buffer and reassign weights to them, thereby breaking the temporal correlation between adjacent samples and improving data utilization efficiency (Figure 4).

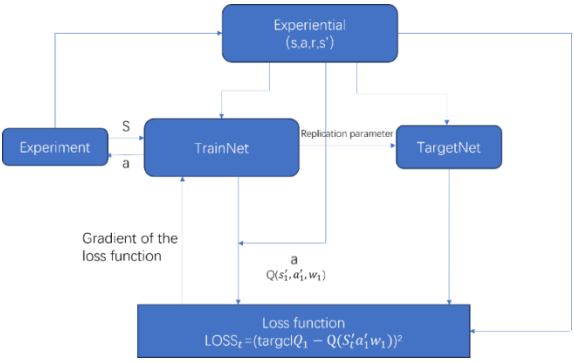

Fig. 4. Schematic diagram of DQN algorithm (Picture credit : Original)

The purpose of DQN's loss function is to reduce the error between the fitted Q value and the actual Q value. The formula is as follows:

$$L(w) = E(s, a, r, s') \sim D[(y - Q(s, a; w)^2] \quad (3)$$

Among them: L(w) is the loss function. w is the weight relationship. Q(s, a, w) is the Q value estimated by the current network, and parameter y is the target Q value. In DeepMind's earliest work [9]. They used DQN to control characters in Atari games, with the most typical example being Breakout. This case demonstrates how deep neural networks can be used to solve high-dimensional visual input problems, while also incorporating prioritized experience replay to improve training efficiency. By combining experience replay and deep neural networks (RNNs), researchers have optimized the performance of intelligent robots in dynamic obstacle avoidance, enabling them to better adapt to high-dimensional inputs (such as visual data) and enhancing their real-time decision-making and dynamic adaptability in complex environments.

3.2 Combination of DQN and Causal Model

Since the Q-learning algorithm does not require prior data, it relies heavily on interactions with the environment and lacks an understanding of the causal relationships within the environment, which may lead to the occurrence of local optima. Introducing causal models can significantly improve this situation.

The core of causal model is to describe the causal relationship between variables, rather than just statistical correlation. It achieves causal relationship modeling and intervention effect calculation through causal graphs and causal inference.

The causal graph is represented using a directed acyclic graph (DAG) to illustrate the causal relationships between variables. Here, S represents the state, A represents the action, and R represents the reward (Figure 5).

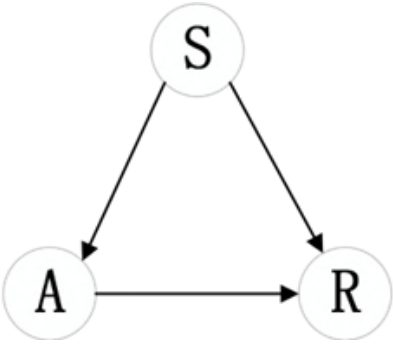

Fig. 5. Cause diagram (Picture credit : Original)

Cause-and-effect diagrams can be used in the following ways:

(1)S → A → R, meaning that the different actions A taken by the intelligent robot in state S will result in different feedback R, which in turn affects the evaluation of the Q-value and consequently impacts the accuracy of the algorithm.

(2)S → R ← A, meaning that the feedback R received by the intelligent robot during interaction with the environment depends on both the state S and the action A executed.

Combining the Q algorithm with causal models can improve sample efficiency, achieving the goal of reducing the exploration sample size through causal inference. Decision-making based on causal relationships is more rational, reducing the misleading effects of correlations. By describing the causal relationships between states, actions, and rewards, the Q-learning algorithm's data efficiency and decision-making accuracy can be significantly improved in complex environments. In fields such as dynamic obstacle avoidance and robot path planning, the causal Q-learning algorithm can learn optimal strategies in complex environments more efficiently. This combination is becoming an important direction in reinforcement learning research [10]. The causal-driven learning framework models the environmental feedback mechanism, enables the agent not only to predict outcomes but also to understand the underlying causal relationships. Compared to traditional black-box reinforcement learning, this integration not only improves the execution efficiency of the current task but, more importantly, constructs a scalable causal cognitive framework, laying the

foundation for subsequent continuous autonomous evolution. Just as upgrading from experiential accumulation to principled cognition, it significantly enhances learning efficiency and decision-making quality.

3.3 Improved Deep Q Network Algorithm Based on Exploring Noise Layer

Recently, in order to improve the exploration ability of intelligent robots, improve the efficiency of sample utilization, reduce dependence on ϵ-greedy strategies, and maintain sufficient exploration of environmental dynamics, researchers have introduced an exploration noise layer in DQN. This has become a new improvement trend in Q-learning algorithms [11].

The principle is: record DQN as Q(s, a, w), and then replace w with $\mu + \sigma + \tau$ to obtain a noisy DQN, recorded as:

$$\widetilde{Q}(s, a, \tau, \mu, \sigma) \cong Q(s, a, \mu + \sigma + \tau) \quad (4)$$

In this context, μ and σ represent the mean and standard deviation, respectively, both of which are parameters of the neural network. They start with initial values and are subsequently learned from experience. τ is the random noise value, with noise intensity dynamically adjusted during training, enabling the agent to escape local optima and explore unvisited state spaces. The introduction of the exploration noise layer also improves sample utilization efficiency, especially in high-dimensional state spaces, reducing dependence on the ε-greedy strategy while maintaining sufficient exploration of the environment dynamics. This improvement upgrades the exploration mechanism of reinforcement learning from "brute-force exhaustive search" to "heuristic perturbation," allowing the agent to maintain curiosity about the unknown while also making rational innovations based on existing knowledge. This mirrors the exploration approach of human scientists: expanding the boundaries of cognition through controlled experiments (σ×τ) within the existing theoretical framework (μ).

In standard DQN, the exploration noise layer is added before the output of the hidden layers or the final layer. Compared to the fixed ϵ-greedy strategy, this approach is more flexible. In complex environments, the noise layer can guide the agent to more quickly discover high-fit regions, thereby avoiding blind exploration and reducing sample waste. In practical dynamic obstacle avoidance scenarios, the DQN improved with noise layers can learn obstacle avoidance strategies more quickly, with the average success rate of avoidance increasing by 10%-20%. This research direction can further be expanded to multi-task learning in the future, using noise layers to enhance cross-task generalization ability. It has broad application potential in fields such as robot control and autonomous driving. In the traditional DQN, the robot requires 200 training episodes to achieve an 80% obstacle avoidance success rate, and collisions still occur occasionally in the later stages due to the fixed ϵ value. After improving DQN with noise layers, it only takes 120 episodes to reach a 92% success rate, with the collision rate approaching zero in the later stages of training, and the σ value automatically decaying to nearly zero.

4 Shortcomings and improvements of existing algorithms

Since the DQN model was proposed, many researchers have demonstrated its superiority, but it still has some shortcomings. Representative examples are listed below and improvement plans are given.

4.1 Deep Recurrent Q Network (DRQN) improves decision delay

The traditional DQN algorithm suffers from the curse of dimensionality in partially observable environments, leading to excessively long convergence times. DQN mainly relies on the ϵ-greedy strategy for exploration, which has high randomness and may waste samples, especially in complex environments. In environments with vast state spaces or sparse rewards, effective exploration can be challenging, and the algorithm may easily fall into local optima. The introduction of deep recurrent Q-networks can address these shortcomings.

DRQN replaces the fully connected layer in the convolutional layer of DQN with an RNN layer, which allows it to remember historical information through its hidden states. DRQN takes the time series of states as input and relies on the historical observation sequence to make more accurate decisions. The DRQN function is as follows:

$$Q_{(O1:\ t,\ a;\ \zeta)=}E[r_1 + \gamma maxQ(o_{1:\ t+1},\ a\ ';\ \zeta'\ ) \qquad (5)$$

Among them, is the parameter of the current Q network, and is the parameter of the target Q network.

Experiments have shown that robots can make decisions more effectively in partially unobservable environments using the DRQN method.

4.2 Improved Deep Q Network Algorithm Based on Exploring Noise Layer

In robot path planning tasks based on visual perception, the value function of state-action pairs differs due to the influence of different actions. However, in some states, the value function is independent of the action. Therefore, by decoupling the action-value function into the state-value function and advantage function, the value evaluation of specific states and their corresponding actions can be made more accurate. The action function is expressed as:

$$Q(s,a,w,\alpha',\beta') = V(s,w,\alpha') + A(s,a,w,\beta')max\ A(s,a',w,\beta') \qquad (6)$$

Where: it is the network parameter of the value function, and the network parameter of the advantage function.

By having each agent independently maintain a DQN network, states and rewards are shared, thereby achieving strategy updates and network updates.

Such a competition mechanism encourages intelligent agents to explore new decision spaces and adjust strategies in real time according to the environment. Through mutual competition, intelligent agents form an equilibrium strategy close to the optimal one.

4.3 Layered Deep Reinforcement to Improve Decision-Making Efficiency

In some complex environments fitting path planning tasks, directly optimizing the strategy based on the final goal is inefficient. Therefore, Hierarchical Reinforcement Learning (HRL) can be utilized to decompose the final goal into multiple subtasks to learn hierarchical strategies, and by combining the strategies of these subtasks, an effective global strategy can be formed.

In some complex target guidance problems, the lack of feedback signals has always hindered the improvement of algorithm performance in spatiotemporal grid space and intrinsic motivation-based hierarchical reinforcement learning. Existing methods like DQN fail to learn effectively in such complex environments. To solve such problems, the ability to perceive hierarchical spatiotemporal abstractions must be developed, and this can be achieved by using certain incentives to promote fitting. Kulkarni et al. proposed an improved hierarchical DQN (Hierarchical Deep Q Network, HDQN). HDQN is a hierarchical DQN based on spatiotemporal abstraction and intrinsic motivation, which layers the value function by setting alphabets at different spatiotemporal scales. The specific model is shown in Figure 6:

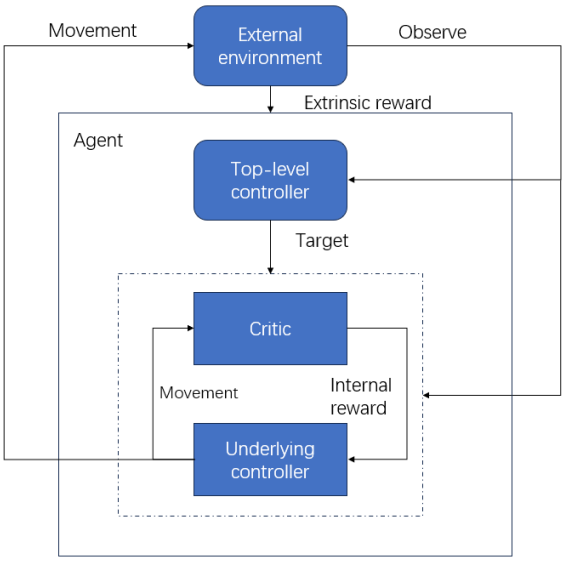

Fig. 6. Hierarchical DQN structure model diagram (Picture credit : Original)

5 Conclusion

This paper analyzes the practical issues faced by robots when interacting with the environment using DQN, as well as the improvement proposals made by researchers both domestically and internationally to address different problems. It also proposes better optimization directions for the existing technical paths. When discussing improvements to the current algorithm, efforts were made to enhance the DQN

algorithm from different technical paths, including the exploration of replay mechanisms, causal systems, and convolutional layer improvements. In the future, to improve the efficiency and stability of the DQN algorithm, it is necessary to focus on data efficiency, environmental interaction capabilities, and decision delay, while making structural improvements to the DQN algorithm.

References

1. G. Z. Tan, H. E. Huan, and A. Sloman, "Ant colony system algorithm for real-time globally optimal path planning of mobile robots," *Acta Automatica Sinica*, vol. 33, no. 3, pp. 279–285, 2007, doi: 10.1360/AAS-007-0279.
2. R. X. Li, L. Fu, L. L. Wang, and X. G. Hu, "Improved Q-learning based route planning method for UAVs in unknown environment," in *Proc. IEEE 15th Int. Conf. on Control and Automation (ICCA)*, Edinburgh, Scotland, Jul. 16–19, 2019, pp. 118–123, IEEE.
3. X. Huang, "Microassembly robot: Key technologies, development, and applications," *CAAI Trans. on Intell. Syst.*, vol. 15, no. 3, pp. 413–424, 2020.
4. C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3, pp. 279–292, 1992.
5. I. Gioroudi, H. Hötzendorfer, N. Koselj, et al., "Development of a microgripping system for handling of microcomponents," *Prec. Eng.*, vol. 32, no. 2, pp. 148–152, 2008.
6. H. Ma and A. Xue, "Research on robot path planning based on deep attention Q network," *Transducer and Microsystem Technologies*, vol. 43, no. 12, pp. 66–70, 75, 2024, doi: 10.13873/J.1000-9787(2024)12-0066-05.
7. X.-N. Kang, "Research on path planning algorithm based on reinforcement learning," Ph.D. dissertation, Yanshan Univ., 2021, doi: 10.27440/d.cnki.gysdu.2021.001844.
8. V. Mnih, K. Kavukcuoglu, D. Silver, et al., "Playing Atari with deep reinforcement learning," arXiv preprint arXiv:1312.5602, 2013.
9. V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015, doi: 10.1038/nature14236.
10. M. Fortunato, M. G. Azar, B. Piot, et al., "Noisy networks for exploration," arXiv preprint arXiv:1706.10295, 2018. (Accessed: Feb. 11, 2025).
11. S. Chen and S. Shen, "An improved path planning method based on reinforcement learning algorithm," *Computer Technology and Development*, pp. 1–8, 2024, doi: 10.20165/j.cnki.ISSN1673-629X.2024.0308. (Accessed: Dec. 27, 2024).