RESEARCH ARTICLE

Ranch robot operation optimization method based on improved deep double q-network algorithm

Qingle Quan[1,*], Jianwei Zhang[1], Hanqing Sun[1], Jiajia Ren[2]

[1]School of Information Engineering, [2]School of Logistics and E-commerce, Henan University of Animal Husbandry and Economy, Zhengzhou, Henan, China.

Received: February 26, 2026; **accepted:** June 10, 2026.

With the rapid advancement of artificial intelligence and agricultural automation, robots have been widely deployed in agricultural production. However, pasture robots currently face high operational error rates due to complex and changeable environments, which limit their work efficiency. To address this issue, this study aimed to optimize ranch robot operations and enhance their operational accuracy and work efficiency. An improved Deep Double Q-Network (DDQN) algorithm was constructed by integrating a sample value evaluation mechanism, a temporary storage (TS) method, and an improved parameter update strategy. The sample value evaluation mechanism dynamically assessed sample importance to prioritize high-value samples during training. The TS unit alleviated performance degradation caused by sample obsolescence, while the improved parameter update strategy enabled adaptive adjustment according to environmental complexity and task difficulty. Simulation experiments were conducted on the OpenAI Multi-Agent Particle Environment (MAPE) platform, comparing the proposed algorithm with MADQN, MADDQN, and MATD3 across three task scenarios of patrol, interception, and herding. The results demonstrated that the improved DDQN achieved success rates of 87.72%, 86.89%, and 86.98% in the three scenarios, respectively, with cumulative average rewards of 76.9, 77.3, and 11.2. Furthermore, the task completion times were 23.1 s, 20.64 s, and 71.36 s, and the average path lengths were 39, 18, and 27, respectively, all superior to the comparison algorithms. The robot driving paths were also smoother and shorter under the proposed algorithm. These findings indicated that the improved DDQN algorithm effectively enhanced the operational accuracy and work efficiency of ranch robots, providing a theoretical foundation and practical guidance for research in agricultural robotics.

Keywords: ranch robot; deep double q-network; agriculture; operational accuracy optimization; sample value evaluation.

[*]**Corresponding author:** Qingle Quan, School of Information Engineering, Henan University of Animal Husbandry and Economy, Zhengzhou, Henan 450000, China. Email: quanqingle@hnuahe.edu.cn.

Introduction

With the swift advancement of artificial intelligence and agricultural automation, robots as intelligent agents are widely used in agricultural production, and their work efficiency is becoming increasingly important [1]. In pasture tasks, robots undertake tasks such as guiding livestock, feeding, and health monitoring [2].

Considerable advances have been achieved in robot control and intelligent algorithms. Eiffert *et al*. introduced a hierarchical planning strategy to enhance the autonomy of farm robots and found that the framework improved the robot's adaptability in the real-world farm environment, but the calculation was relatively complex [3]. To solve the path planning (PP) problem of outdoor farm robots, Qi *et al*. proposed a field robot PP

algorithm grounded in the improved A* algorithm and the artificial potential field algorithm. The results demonstrated that the algorithm was better than the traditional A* algorithm [4]. Ge *et al*. developed a positioning method combining the iterative nearest point algorithm and the extended Kalman filter to tackle the challenge of inaccurate feeding positioning of herding robots and found that the approach improved the positioning accuracy of the robot [5]. Gunathilaka *et al*. proposed a robot PP algorithm grounded in the improved Agora Fielek algorithm to solve the problem of inaccurate navigation of robots in unstructured terrain and found that the algorithm was effective, but the computational efficiency was low [6]. To boost the accuracy of data prediction by the grass perception robot, Zhang *et al*. proposed a multidimensional PSO algorithm. The approach reduced the robot's time consumption, but the generalization ability was not verified [7]. Deep Double Q-Network (DDQN) is a reinforcement learning (RL) algorithm that effectively alleviates the overestimation problem of Deep Q-Network (DQN) by introducing a double network structure and improves the learning stability and accuracy. It has been widely used in robot control and intelligent control [8, 9]. The DDQN algorithm has been applied to multiple fields. Ni *et al*. introduced an improved DDQN algorithm aimed at addressing the issues of sparse rewards and the failure to differentiate sample importance in robot PP. The results demonstrated that the algorithm shortened the path length (PL) by 11.5% [10]. In view of the problem that the current stock trading decision model has low accuracy in capturing complex features of stock trends, Cui *et al*. proposed a DDQN-based model and reported that the model showed superior performance in multiple datasets [11]. Zhang *et al*. developed a multi-objective RL algorithm based on DDQN to tackle the challenge of minimizing latency and energy consumption during the computation offloading process while ensuring data privacy and edge resource load balancing in mobile edge computing environment. The results demonstrated that the algorithm improved the overall energy consumption by 30.24% compared with the traditional RL method [12]. To address the problem of low efficiency in updating target network (TN) data in driving control algorithms, Shi *et al*. proposed an adversarial DDQN algorithm and found that the success rate (SR) of this algorithm was 6% higher than that of the traditional DQN algorithm [13]. Yuan *et al*. introduced a real-time scheduling method grounded in improved DDQN to solve the scheduling problem of flexible workshops with automated guided vehicles and found that the approach was effective and accurate [14]. Although many control methods have been applied to robots, due to the complex and changeable pasture environment, robots face high requirements for accurate navigation and efficient execution of grazing tasks during operation [15]. There are still problems such as high operation error rate [16]. Furthermore, the conventional DDQN algorithm suffers from defects such as slow convergence speed and performance degradation caused by sample obsolescence, which limits its training efficiency and stability in dynamic pasture environments.

This study introduced the DDQN algorithm to optimize the operation of a ranch robot and improved the experience replay (ER) buffer using sample value evaluation and temporary storage (TS) methods to increase its training speed. Simultaneously, an improved parameter update strategy was used to refine its periodic update strategy, enhancing its adaptability and operational accuracy in complex environments to construct an improved DDQN algorithm for ranch robot operation optimization. This research enhanced the operational accuracy and work efficiency of pasture robots, thereby reducing the labor intensity of herders and labor costs. Furthermore, it provided a theoretical foundation and practical guidance for related research in agricultural robotics, contributing to the advancement of intelligent automation in pasture management.

Materials and methods

Basic DDQN algorithm and its improved empirical replay buffer

Through the workflow of DDQN algorithm, the agent first engaged in interactions with its surrounding environment, performed actions, and collected information. The information collected was then stored within the ER buffer, a key component in RL, improving learning efficiency, stability, and generalization ability. By storing and replaying the agent's historical experience, it rendered the learning process both more efficient and more stable. The agent randomly drew samples from the ER buffer to break temporal link and improve sample efficiency. The EN selected the optimal action for the next state according to the current policy. TN was employed to compute the target Q-value. Subsequently, the agent calculated the error between the EN's output and the target value and used the error to update the parameters of the EN through gradient descent. The agent periodically copied the parameters of the EN to the TN to ensure the stability of the TN, thereby achieving efficient learning and decision-making in intricate environments. The parameter update method $G(\theta)$ of the EN was represented below.

$$G(\theta) = E[(r + \gamma Q(s_{t+1}, s_t; \theta^{'}) - Q(s, a; \theta))^2] \quad (1)$$

where s and a were the agent's current state and action. $s^{'}$ and $a^{'}$ were the next state and action. r was the reward after performing the action. θ and $\theta^{'}$ were the parameters of the evaluation network (EN) and the TN. Q was the action value function. γ was the discount factor. E was the expectation operator. $Q(s, a; \theta)$ was the Q value estimate of the action $a^{'}$ in the current state s. t was time. s_t and s_{t+1} were the states at time t and $t+1$. Parameter update improved the agent's decision-making ability in the environment by minimizing the squared error between the EN output and the target value. The parameter adjustment process was presented as follows.

$$Q(s_t, a_t; \theta) \leftarrow Q(s_t, a_t; \theta) + \alpha[r_t + \gamma Q(s_{t+1}, a^{'}; \theta^{'}) - Q(s_t, a_t; \theta)] \quad (2)$$

where $a^{'}$ was the next action. $Q(s, a^{'}; \theta)$ was the EN's value assessment of the state s and action $a^{'}$. $Q(s^{'}, a^{'}; \theta^{'})$ was the target's value assessment of the state $s^{'}$ and action $a^{'}$. α was the learning rate. The DDQN algorithm's ER buffer used random sampling when extracting data, which involved continuous saving and deleting operations, thus reducing the training speed [17, 18]. Therefore, this study improved the ER buffer by employing a sample value evaluation method and a TS method. The improved ER buffer process involved the agent and environment interacting to generate sample data. This generated sample data was then stored within a TS unit. The samples stored in the TS unit were then assigned priority based on the time difference error, which helped the algorithm learn from samples with important information, thereby improving the algorithm's learning ability. The calculation of the time difference error was expressed below.

$$\sigma = r + \gamma Q(s^{'}, a^{'}; \theta^{'}) - Q(s, a; \theta) \quad (3)$$

where σ was the time difference error. $Q(s, a; \theta)$ was the EN's assessment of the value of state s and action a. Next, samples were selected into the ER buffer according to the probability $p(i)$, which could be expressed as follows.

$$p(i) = \frac{(e_i + \beta)^{\chi}}{\sum_{j=1}^{k} (e_j + \beta)^{\chi}} \cdot \frac{log(\phi + m_i)}{\sum_{j=1}^{k} (\phi + m_i)} \quad (4)$$

where e_i was the temporal difference error of sample i. k was the total number of samples in the TS unit. χ was the adjustment coefficient. m_i was the number of times sample i being selected. β and ϕ were minimal constants used to ensure that the probability was non-zero. Simultaneously, the total amount of samples in the ER buffer was limited to 0.7 times the number of TS units, and the remaining samples were cleared. Then, based on the value of the sample, the probability qp was periodically used to determine whether to eliminate sample i in the

ER buffer. The calculation of qp was expressed as follows.

$$qp(i) = \frac{exp(-\eta \cdot (y_i + \mu)^w)}{\sum_{i=1}^{k} exp(-\eta \cdot (y_i + \mu)^w)} \quad (5)$$

where w was the adjustment factor. η was the proportionality constant. y_i was the initial priority of the sample. μ was a very small constant. Finally, to ensure the diversity of the sample, sample i was selected by balancing the probability pp , and the calculation of pp was shown below.

$$pp = \frac{1}{\frac{z}{y_i + \varepsilon} + \frac{1 - z}{n + \varepsilon}} \quad (6)$$

where n was the average priority of all samples. z was the weight coefficient. ε was a constant that restricted the probability of y_i to be too high when the value of y was particularly small.

Improved DDQN construction for ranch robot operation control

To address the issue of overestimating state value during updates in DDQN, this study proposed an improvement to its network parameter update mechanism. The current network parameters, TN parameters, and ER buffer were initialized. A batch of samples was then randomly chosen from the ER buffer. The agent chose an action according to the current network hyperparameters and executed it. Simultaneously, rewards and the next state were collected and were then added to the ER buffer and updated. The target value was calculated by employing the current TN parameters, and the current network parameters were updated by using the batch of samples. The TN parameters were periodically updated by using the current network parameters. Through iterative iteration, the above steps were repeated until the max number of learning rounds was reached, at which point the current and TN parameters were output. This update process was represented as follows.

$$Q(s_t, a^{'}; \theta) \leftarrow Q(s, a^{'}; \theta) + \alpha(r + \gamma * Q(s^{'}, \arg\max Q(s^{'}, a; \theta^{'}) - Q(s, a; \theta)] \quad (7)$$

An improved DDQN algorithm was built based on the above content. The working process of the improved DDQN included that the parameters of the max iteration count and the agent's exploration rate ξ were first initialized. Meanwhile, the network parameters of the TN and the EN were initialized, and the two networks had the same structure. In addition, the ER buffer and the TS unit were initialized. Next, the iteration was performed, and the environment and the original state of the agent were initialized. Meanwhile, a random number was generated. If it was less than ξ , the current best action was selected according to the EN. Otherwise, the current best action was randomly obtained. The agent executed the best action to obtain the current reward value and the latest state information. The information obtained was saved to the TS unit. When the TS unit was full, the value was evaluated by using equation (3). At the same time, the sample was extracted from the ER buffer according to the probability calculated by equation (4). After extraction, the value was evaluated by using equation (3). After evaluation, the sample was selected according to the probability calculated by equation (5) and saved to the ER buffer. After saving, the TS unit was cleared. When the algorithm entered the training phase, the sample value in the ER buffer was calculated by using equation (3). Samples were collected based on the priority calculated according to equation (6). The loss value of the sample was calculated by using equation (2). Finally, the EN parameters were updated by using equation (1). When the TN reached the update cycle, the EN was used to update it.

Design of a ranch robot control method based on improved DDQN

After constructing the improved DDQN algorithm, the study aimed to achieve robot control in pasture task scenarios, taking common

pasture patrol, livestock interception, and livestock driving task scenarios as examples. The robot patrolled point-by-point within a pre-defined map area. Simultaneously, it used sensors and cameras to collect information such as the location of the cattle herd and the pasture environment within the area. The robot intercepted the cattle herd within its patrol area to prevent them from running out of the designated map range. During this process, the robot used its own strategy rules to move in front of the livestock and guide them back into the area using sound signals and visual cues. The robot used sound or frequency signals to drive the cattle herd to the designated target location. Based on the above scenarios, a reward function $JL(i,t)$ was set for each robot and was expressed as follows.

$$JL(i,t)=\sum_{i=1}(\alpha_i\cdot\frac{S_{cover,i}(t)}{S_{total}})+\delta\cdot(1-\frac{b_{i,cattle}(t)}{b_{max}})+\psi\cdot coll(i,t)+\kappa\cdot jc(i,t)+(-\tau\cdot U(i,t)) \quad (8)$$

where $S_{cover,i}(t)$ was the size of the area reached by the agent i at time t. κ was the coefficient for detecting predators. τ was the penalty coefficient for unnecessary movement by the agent. $b_{i,cattle}(t)$ was the distance between agent i and herd. b_{max} was the maximum allowable monitoring distance. ψ was the collision penalty coefficient. $coll(i,t)$ was the number of collisions by the agent i at time t. $jc(i,t)$ was the number of predators detected by agent i at time t. δ was the reward coefficient for approaching the herd. S_{total} was the total area of the target area. $U(i,t)$ was the amount of movement by agent i considered unnecessary at time t. The reward function $LJ(t)$ for the interception task scenario could be expressed as follows.

$$\begin{aligned}LJ(t)=&TJ\cdot JL^{-cd_{ljjl}}+KJ\cdot\frac{1}{WX+CS}+(-PX\cdot\left\|\vec{a}_t-\vec{a}_{t-1}\right\|^2)\\&+\upsilon\sum_{j\neq i}\frac{\vec{a}_i\cdot\vec{a}_j}{\left\|\vec{a}_i\right\|\left\|\vec{a}_j\right\|}+(-BM\cdot\sum_{i\neq j}\frac{1}{(o_{ij}+\beta)^2})+WC\cdot 1\end{aligned} \quad (9)$$

where TJ was the coefficient for adjusting the reward size. c was the decay rate. d_{ljjl} was the distance between the interception point and the herd. KJ was the reward adjustment coefficient. CS was a constant to ensure that the denominator did not equal zero. PX was the coefficient used to adjust the degree of influence of the smoothness reward. $\vec{a}_t$ and $\vec{a}_{t-1}$ werere the action vectors of the agent at the current time and the previous time, υ is the cooperative reward coefficient, $\vec{a}_i$ and $\vec{a}_j$ were the action vectors of different agents. BM was the penalty coefficient. o_{ij} was the distance between any robot and the herd. WC was the completion reward coefficient. The reward function $QG(t)$ for the herding task scenario could be expressed below.

$$\begin{aligned}QG(t)=&J_{QG}\cdot cos\frac{\pi\cdot d_{cent\_to\_target}}{2d_{qg\_max}}+JM_{qg}(1-tanh(var(d_{cattle\_to\_cenrold}))\\&+(-YP_{qg}\cdot log\frac{S(t)}{\partial_{avg}+\beta})+\beta\cdot e^{-\frac{path\_opti}{zd_{jl}}}+h_{qg}\cdot\left|\sum_{j\neq i}\frac{\vec{\partial}_i\cdot\vec{\partial}_j}{\left|\vec{\partial}_i\right|\left|\vec{\partial}_j\right|}\right|\\&+AQ_{qg}(1-\frac{\delta_{speed}}{SD_{max}})+W_{qg}(\frac{T_{max}-T_{act}}{T_{max}})\end{aligned} \quad (10)$$

where J_{QG} was the target area proximity reward coefficient. $path\_opti$ was the trajectory deviation degree. $d_{cent\_to\_target}$ was the distance from the herd to the target area. $\vec{\partial}_i$ and $\vec{\partial}_j$ were the positional velocity vectors. d_{qg_max} was the maximum considered distance. JM_{qg} was the herd density reward coefficient. $d_{cattle\_to\_cenrold}$ was the distance between the herd and the robot. YP_{qg} was the dynamic penalty adjustment coefficient to avoid oppression. $S(t)$ was the oppression index. ∂_{avg} was the robot's average speed. zd_{jl} was the maximum travel length during task execution. AQ_{qg} was the adjustment coefficient to adjust the safe driving reward. δ_{speed} was the standard deviation of the agent's speed. h_{qg} was the adjustment factor for system cooperation reward. SD_{max} was the agent's maximum speed. W_{qg} was the reward coefficient. T_{max} and T_{act} were the maximum allowed time of the task and the actual time required to complete the task, respectively.

Experimental parameters and environmental settings

After constructing the improved DDQN algorithm and reward functions for different task environments, simulation experiments were conducted using the OpenAI platform's Multi-Agent Particle Environment (MAPE) (https://github.com/openai/multiagent-particle-envs). Multi-Agent Deep Q-Network (MADQN), Multi-Agent Double Deep Q-Network (MADDQN), and Multi-Agent Trust Region Policy Optimization (MATD3) based on the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) framework (https://github.com/openai/maddpg) were employed for comparison studies with modifications to the network architecture to construct the Q-learning variants. Experimental metrics included average reward, SR, average PL, and task time. The experimental parameters were configured. The discount factor was set to 0.9, and the batch size was 128. The learning rates for TN and EN were 0.01 and 0.006, respectively. The size of the deceleration zone based on experience was adjusted to 1×10^5. The number of nodes in both the first and second hidden layers was 32. The initial exploration variance was 0.4, and the maximum driving distance of the intelligent agent was 80. The number of training rounds was set to 1×10^4. The probability of choosing an action in the experience back-buffering zone was 0.89. The noise variance of the target action of the agent was 0.01. The TN update interval was 2, and the number of agents was 3. The simulation data were generated through the interaction between agents and the environment during the training process. The dataset consisted of state-action-reward-next state tuples collected in real-time. Each episode generated interaction data with a maximum of 500 timesteps, and a total of 10,000 training episodes were executed for each scenario. The data types included agent position coordinates, velocity vectors, distance to targets, collision status, herd location information, and reward values. The simulation environment was configured as a 2D pasture map with dimensions of 100 m × 100 m, containing three intelligent agents and multiple target waypoints corresponding to patrol points, interception boundaries, and herding destinations. All data were collected through simulation rather than field measurement, ensuring reproducibility and controlled experimental conditions. The experiments were conducted on a computer equipped with an Intel Core i9-13900K processor operating at 5.8 GHz, 32 GB of RAM, and 500 GB of storage capacity. The operating system was Windows 10 64-bit, and the software version was Matlab 2021a (https://www.mathworks.com/products/matlab.html) .

Statistical analysis

The data collected from different algorithms and scenarios were statistically analyzed using the Wilcoxon rank-sum test for the differences between the improved DDQN and all comparison algorithms. *P* value less than 0.05 was defined as statistically significant difference, while *P* value less than 0.01 was defined as very significant difference.

Results and discussion

Performance analysis of the improved DDQN

(1) Cumulative average reward comparison

The performance of proposed improved DDQN was compared with that of the other algorithms in different scenarios under MAPE. In the patrol mission scenario, the proposed improved DDQN algorithm had a cumulative average reward of 76.9, significantly higher than MADDQN's 17.8, MATD3's -26.7, and MADQN's -33.6 (Figure 1a). In the interception mission scenario, the cumulative average rewards of the improved DDQN, MADDQN, MATD3, and MADQN algorithms were 77.3, 61.2, 3.7, and -5.6, respectively. Among them, the proposed improved DDQN algorithm had the highest cumulative average reward (Figure 1b). In the expulsion mission scenario, the proposed improved DDQN algorithm had the highest cumulative average reward with a value of 11.2 (Figure 1c). There were significant differences between the improved DDQN and all comparison algorithms in all three scenarios ($P < 0.01$). The

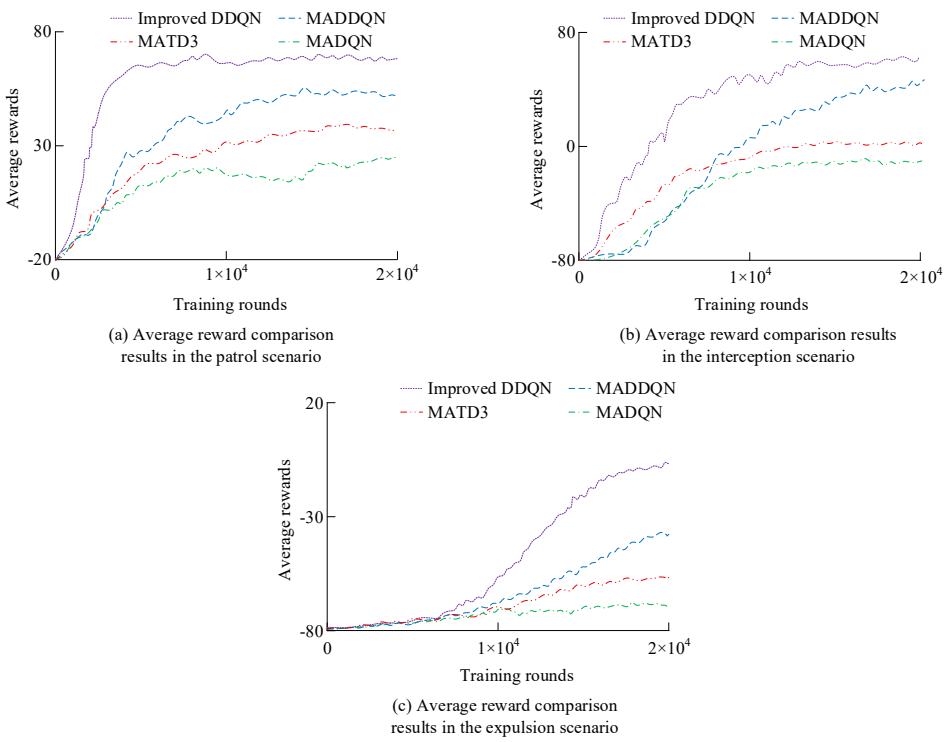

Figure 1. Comparative results of the average rewards for each algorithm in various scenarios.

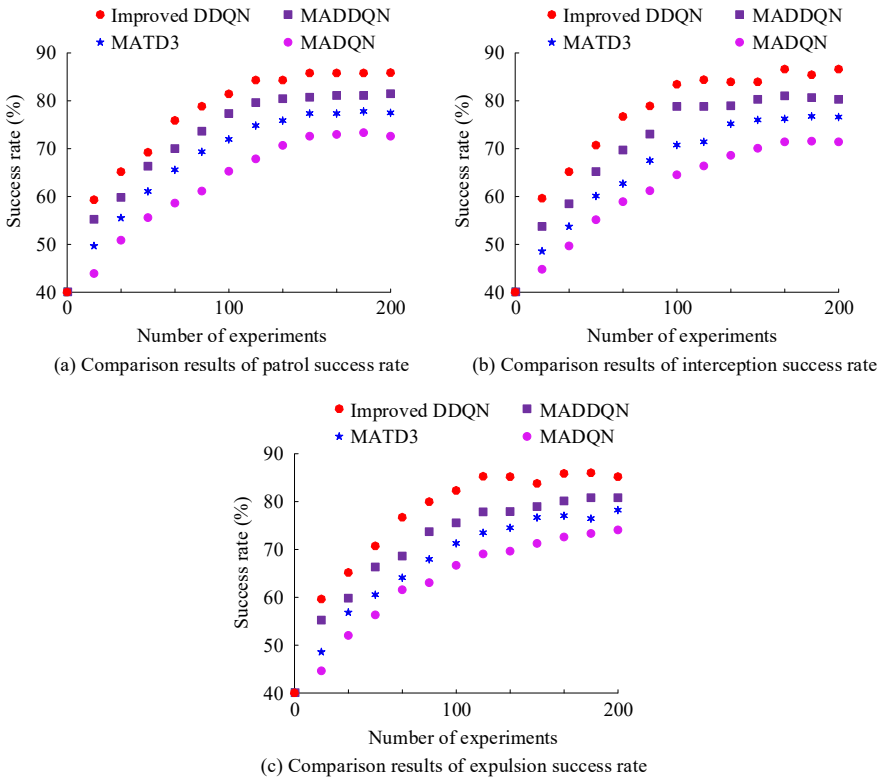

Figure 2. Comparison results of SRs of each algorithm in various task scenarios.

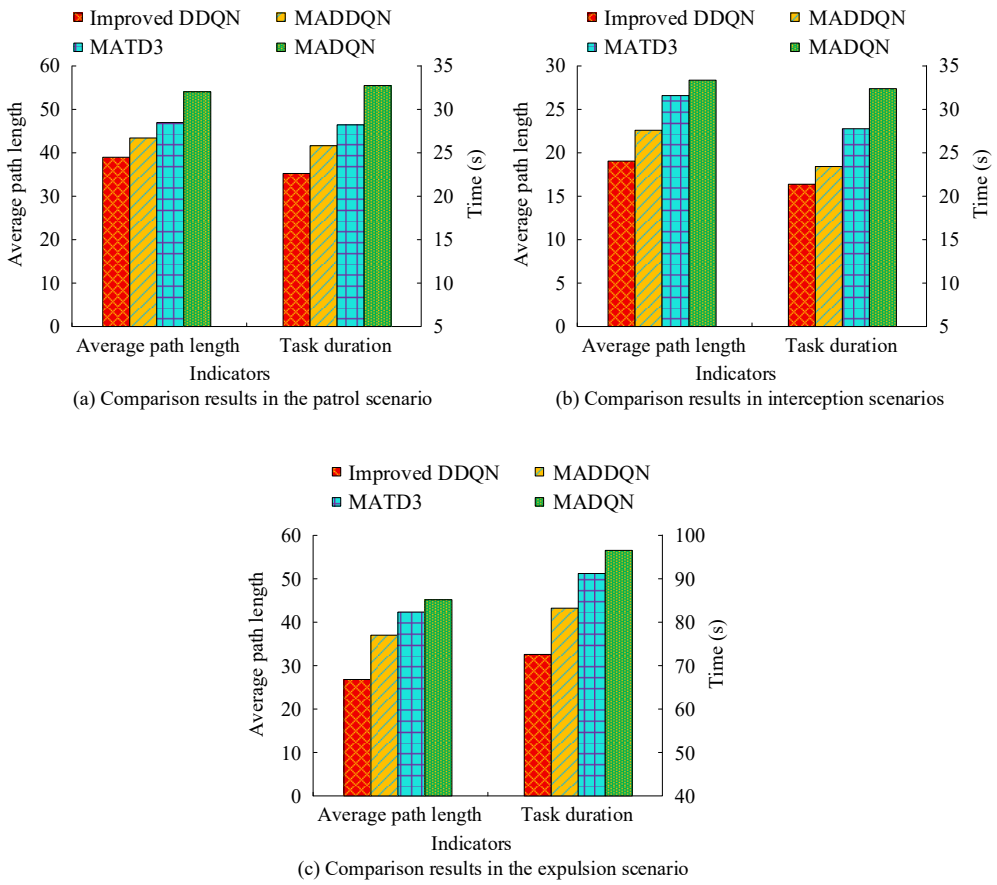

Figure 3. Comparison results of average PL and TCT under various task scenarios.

results indicated that the introduced improved DDQN outperformed the compared algorithms on cumulative average reward.

(2) Success rate comparison

The SR comparison results of each algorithm in different mission scenarios showed that the SRs of the proposed improved DDQN algorithm were 87.72%, 86.89%, and 86.98% in patrol, interception, and deflection scenarios, respectively, significantly higher than the 81.23%, 80.06%, and 79.96% of the MADDQN algorithm ($P < 0.01$). Furthermore, it surpassed 78.21%, 76.53%, and 74.86% of the MATD3 algorithm and 72.68%, 71.22%, and 74.16% of the MADQN algorithm ($P < 0.01$) (Figure 2). These results demonstrated that the introduced algorithm exhibited the best performance in terms of SR compared to the comparative algorithms.

(3) PL and task completion time (TCT) comparison

The comparison of average PL and (TCT) for each algorithm in different task scenarios demonstrated that in the patrol scenario, the average PL and TCT of the proposed improved DDQN algorithm were 39 and 23.1 s, while the average PL and TCT of the MADDQN, MATD3, and MADQN were 41 and 26.3 s, 47 and 28.6 s, and 56 and 33.28 s, respectively. Among these, the proposed improved DDQN had the shortest average PL and TCT ($P < 0.05$) (Figure 3a). Furthermore, the average PL of the proposed DDQN algorithm in the interception and deflection scenarios were 18 and 27, respectively, and the TCT was 20.64 s and 71.36 s, respectively, all lower than the MADDQN, MATD3, and MADQN algorithms ($P < 0.05$) (Figures 3b and 3c). These results indicated that, in terms of average PL and TCT, the proposed

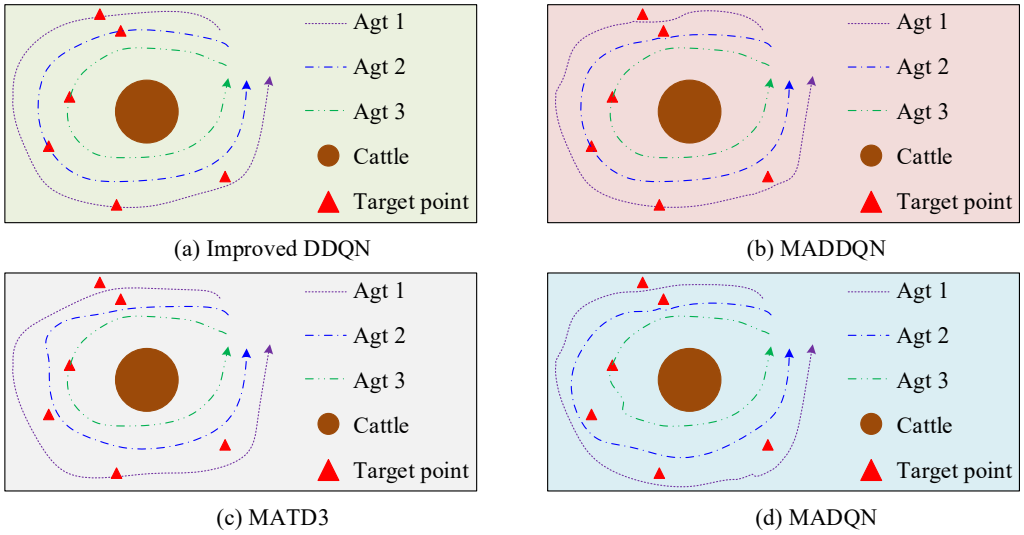

Figure 4. Route comparison results of the lower robots under the patrol task scenario.

improved DDQN algorithm outperformed the compared algorithms.

Analysis of operational performance under different task scenarios

After training convergence, the improved DDQN and comparison algorithms were executed to generate robot driving paths in three task scenarios. The position coordinates of agents 1, 2, and 3 were recorded at each timestep and visualized to compare path smoothness and target positioning accuracy under different algorithms. In the multi-agent system, three intelligent agents (Agt 1, Agt 2, and Agt 3) were deployed to perform collaborative tasks. Agt 1, Agt 2, and Agt 3 represented the three grazing robots in the patrol, interception, and herding scenarios, respectively. To further verify the capability of the introduced algorithm, it was run alongside a comparison algorithm in different scenarios, and the performance was analyzed using the travel route as an indicator. The results showed that, under the proposed improved DDQN algorithm, the robot's patrol path passed through the target points relatively well, and the travel curve was relatively smooth (Figure 4a). Under the MADDQN algorithm, although the robot's patrol route mostly passed through the target points, the travel curve was more tortuous than proposed algorithm, and the deviation from some target points was larger (Figure 4b). Under the MATD3 and MADQN algorithms, the robot's patrol path deviated even further from the target points, and the travel curve fluctuated more (Figures 4c and 4d). Therefore, in patrol scenarios, the proposed DDQN algorithm showed better performance compared to the comparative algorithms. The comparison results of the robot's routes in the interception task scenario showed that, in the interception task scenario, the proposed improved DDQN algorithm had the shortest robot path distance, the highest smoothness, and the most stable operation in the robot's driving trajectory task scenario compared to the MADDQN, MATD3, and MADQN algorithms (Figure 5). The comparison results of the robot's path in the driving task scenario demonstrated that, in the herding task scenario, the proposed improved DDQN algorithm achieved a stable travel path and accurate target positioning compared to the other algorithms that were tortuous and fluctuated significantly (Figure 6). The result indicated that, compared to the comparative algorithms, the proposed improved DDQN algorithm could better ensure the robot completing the herding task in the shortest possible time.

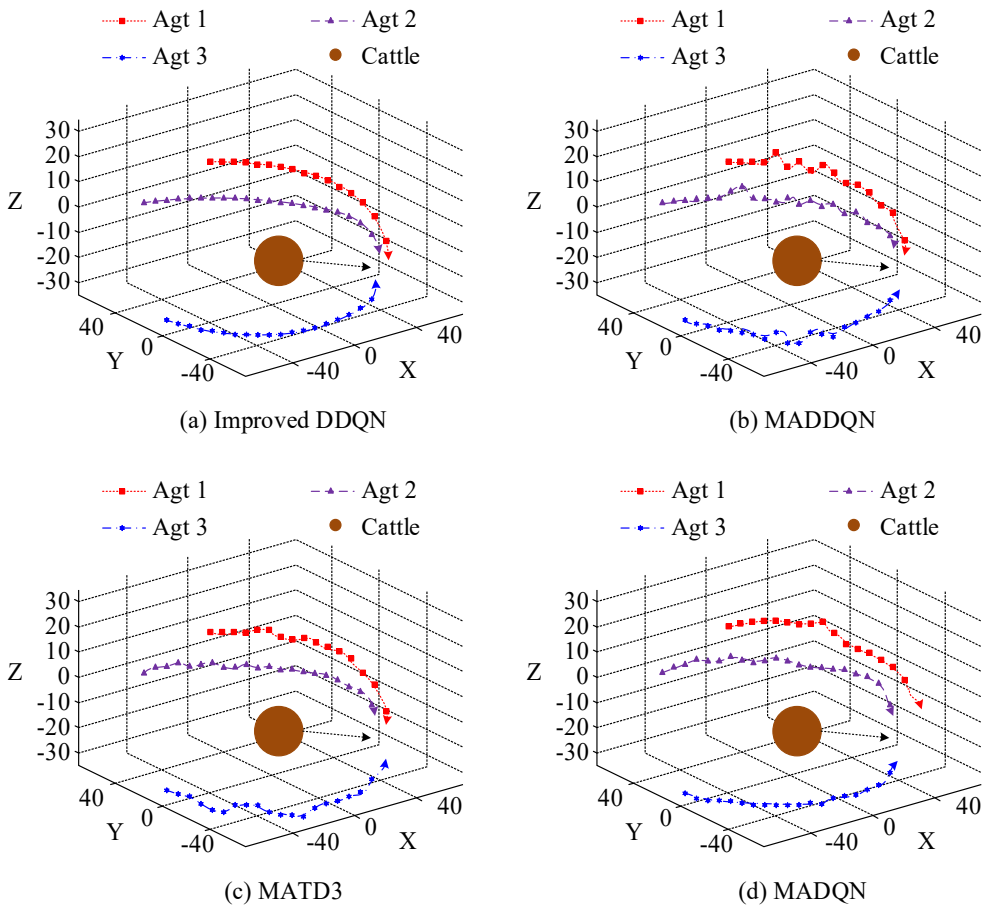

Figure 5. Results of the robot's route in the interception mission scenario.

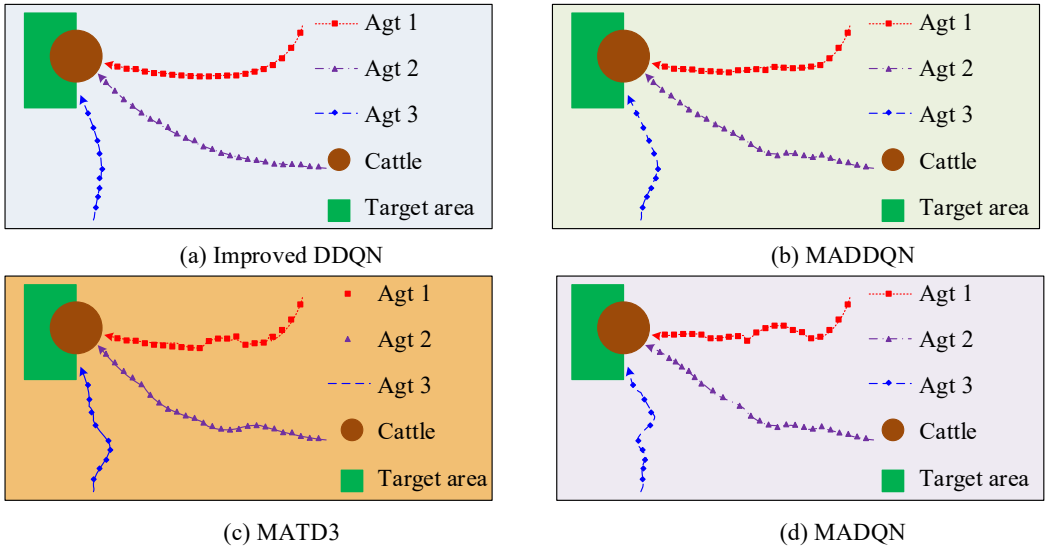

Figure 6. Driven task scenarios robot path comparison results.

Discussion

This study compared the improved DDQN algorithm's performance through experimental analysis. The improved DDQN had significant advantages in success rate, cumulative average reward, PL, and TCT. In the success rate comparison, the improved DDQN achieved higher results in patrol, interception, and herding scenarios than the comparison algorithms, which were similar to the DDQN results reported by Yin *et al*. who introduced average Q-value estimation and reward redistribution to improve decision accuracy in robot PP [19]. In the cumulative average reward comparison, the improved DDQN achieved the results in the three scenarios all superior to MADDQN, MATD3, and MADQN, which coincided with the improved DDQN results proposed by Lin and Wen, who demonstrated superior reward accumulation through optimized network architecture [20]. In the PL comparison, the average PLs of the improved DDQN in patrol, interception, and herding scenarios were all shorter than the comparison algorithms. This result was similar to the PP result derived from Kobayashi and Kawamura, who achieved efficient PL reduction through dynamic task reassignment [21]. In addition, the TCTs of improved DDQN in the three scenarios were all superior to the comparison algorithms. The result was consistent with the trajectory planning results proposed by Chai *et al*., who demonstrated accelerated convergence and efficient task execution in unknown environments [22]. In the path trajectory comparison, the improved DDQN achieved smooth driving paths and accurate target positioning in all three scenarios with the shortest path distance, highest smoothness, and most stable operation. The results indicated that the improved DDQN enhanced the robot's adaptability to complex environments. Furthermore, the proposed model achieved efficient task completion, low collision rates, and stable cooperative behavior, which indicated that the ranch robot operation optimization model based on improved DDQN accurately and effectively completed patrol, interception, and herding tasks in complex pasture environments. However, this study still had some limitations. The experiments were conducted in a simulation environment, and the performance of the improved DDQN in real-world pasture conditions might be affected by unpredictable factors such as weather, terrain variations, and animal behavior dynamics. In addition, the current reward functions were designed for specific task scenarios, and their generalization to other pasture tasks required further validation. Furthermore, the algorithm's computational complexity increased with the number of agents, which might limit its scalability for large-scale ranch operations. In future research, field experiments should be conducted to validate the algorithm's effectiveness in real pasture environments. Additionally, the integration of image enhancement algorithms such as graph neural networks should be explored to improve perception capabilities of robots in complex visual conditions. This study introduced an improved DDQN algorithm to optimize ranch robot operations. By integrating a sample value evaluation mechanism, a TS method, and an improved parameter update strategy, the algorithm enhanced training speed, decision-making stability, and adaptability in complex pasture environments. The improved DDQN algorithm effectively improved the operational accuracy and work efficiency of pasture robots, providing a theoretical foundation and practical guidance for research in agricultural robotics.

Acknowledgements

The research was supported by 2023 Henan Provincial Key R&D and Promotion Special Project (Scientific and Technological Breakthrough) (Grant No. 232102210101).

References

1. Waqas M, Naseem A. 2025. Artificial intelligence in sustainable industrial transformation: A comparative study of Industry 4.0 and Industry 5.0. FSI. 1(1):A2.

2. Bretas IL, Dubeux JCB, Cruz PJR, Oduor KT, Queiroz LD, Valente DS, *et al*. 2024. Precision livestock farming applied to grazingland monitoring and management — A review. Agron J. 116(3):1164-1186.
3. Eiffert S, Wallace ND, Kong H, Pirmarzdashti N, Sukkarieh S. 2022. Resource and response aware path planning for long-term autonomy of ground robots in agriculture. Field Robot. 2(1):1-33.
4. Qi H, Jiang J, Wang C. 2024. Path planning for agricultural robots in wild livestock farm environments. Int J Agric Biol Eng. 17(4):207-216.
5. Ge K, Sun Y, Niu Z, Cheng W. 2025. Autonomous navigation push material robot in ranch based on multisensor. Proc SPIE. 13692:262-267.
6. Gunathilaka WMD, Kahandawa G, Ibrahim MY, Hewawasam HS, Nguyen L. 2025. Agoraphilic-3D: A novel algorithm for robot path planning in mapless uneven terrains. IEEE Access. 13:193802-193819.
7. Zhang Y, Wang T, You Y, Wang D. 2024. Optimization design method for typical grassland perception robot system. Memet Comput. 16(4):563-586.
8. Ji Y, Wang Y, Zhao H, Gui G, Gacanin H, Sari H, *et al*. 2023. Multi-agent reinforcement learning resources allocation method using dueling double deep Q-network in vehicular networks. IEEE Trans Veh Technol. 72(10):13447-13460.
9. Li M, Yang Z. 2024. Double deep Q-network-based task offloading strategy in mobile edge computing. Proc SPIE. 13224:290-298.
10. Ni P, Mao P, Wang N, Yang M. 2025. Robot path planning based on improved A-DDQN algorithm. J Syst Simul. 37(9):2420-2430.
11. Cui K, Hao R, Huang Y, Li J, Song Y. 2023. A novel convolutional neural networks for stock trading based on DDQN algorithm. IEEE Access. 11:32308-32318.
12. Zhang X, Wu W, Zhao Z, Wang J, Liu S. 2023. RMDDQN-learning: Computation offloading algorithm based on dynamic adaptive multi-objective reinforcement learning in internet of vehicles. IEEE Trans Veh Technol. 72(9):11374-11388.
13. Shi P, Zhang J, Hai B, Zhou D. 2024. Research on dueling double deep Q network algorithm based on single-step momentum update. Transp Res Rec. 2678(7):288-300.
14. Yuan M, Zheng L, Huang H, Zhou K, Pei F, Gu W. 2025. Research on flexible job shop scheduling problem with AGV using double DQN. J Intell Manuf. 36(1):509-535.
15. Poojari M, Hanumanthappa H, Prasad CD, Jathanna HM, Ksheerasagar AR, Shetty P, *et al*. 2024. Computational modelling for the manufacturing of solar-powered multifunctional agricultural robot. Int J Interact Des Manuf. 18(8):5725-5736.
16. Azmi HN, Hajjaj SSH, Gsangaya KR, Sultan MTH, Mail MF, Hua LS. 2023. Design and fabrication of an agricultural robot for crop seeding. Mater Today Proc. 81(1):283-289.
17. Shuhai J, Shangjie S, Cun L. 2024. Path planning for outdoor mobile robots based on IDDQN. IEEE Access. 12:51012-51025.
18. Zhai H, Zhou X, Zhang H, Yuan D. 2024. Delay minimization in hybrid edge computing networks: A DDQN-based task offloading approach. IEEE Trans Veh Technol. 73(10):15098-15108.
19. Yin Y, Zhang L, Shi X, Wang Y, Peng J, Zou J. 2024. Improved double deep Q network algorithm based on average Q-value estimation and reward redistribution for robot path planning. Comput Mater Contin. 81(2):2769-2790.
20. Lin Y, Wen J. 2023. Improved dueling deep Q-networks based path planning for intelligent agents. Int J Veh Des. 91(1-3):232-247.
21. Kobayashi T, Kawamura T. 2024. Flexible path planning for multi-agent field observation. J Robot Mechatron. 36(5):1262-1272.
22. Chai R, Niu H, Carrasco J, Arvin F, Yin H, Lennox B. 2022. Design and experimental validation of deep reinforcement learning-based fast trajectory planning and control for mobile robot in unknown environment. IEEE Trans Neural Netw Learn Syst. 35(4):5778-5792.