Actor–Critic Deep Reinforcement Learning for Multi-Objective Intelligent Irrigation Scheduling: Algorithm and Edge-Cloud Management System

Peng Huang
City University of Hong Kong, Hong Kong Special Administrative Region, 999077, China
E-mail: hxjhjt@126.com

Keywords: Deep reinforcement learning, agricultural irrigation, intelligent scheduling, multi-objective optimization

Received: August 26, 2025

Against the backdrop of increasingly prominent climate fluctuations and water scarcity, the demand for precision and intelligence in agricultural irrigation continues to rise. This article focuses on the research of "agricultural irrigation intelligent scheduling algorithm and management system based on deep reinforcement learning", aiming to construct a technical solution that combines decision-making adaptability and resource utilization efficiency. At the algorithmic level, a deep reinforcement learning model is constructed using an improved DQN combined with policy gradient fusion, ensuring consistency between algorithm description and system implementation to map multimodal data such as soil moisture, evapotranspiration, and meteorological predictions collected by field sensing networks into state representations in the irrigation strategy space. The strategy function is optimized using the Time Difference (TD) method to enable the system to continuously update decisions in a dynamic environment. In order to avoid the limitations of single objective optimization, a multi-objective reward function was designed, which integrates crop yield, water resource utilization rate, and energy consumption into the evaluation indicators, and achieves adaptive balance through normalization and weight adjustment. At the system implementation level, a management platform integrating data collection, edge computing, cloud decision-making and mobile visualization is built to support the automatic generation, real-time adjustment and historical data backtracking analysis of irrigation plans. Field trials on a 35-ha wheat–corn site (12 plots, 4 months) evaluated a DQN–Policy Gradient hybrid, trained for 5000 episodes (200 steps each) with lr=0.0005, batch size=64, and buffer=10,000. Rewards weighted efficiency (0.5), yield (0.3), and energy (0.2).The system achieved 88.1% ± 1.7% water use (n=30, p<0.01), representing a 12.7% improvement in water resource utilization, and 8.3% ± 1.2% yield gain (n=30, p<0.05), outperforming thresholds.The research results provide a scalable technical path for intelligent management of agricultural water conservancy, and provide practical verification for the application of deep reinforcement learning in complex resource scheduling scenarios.

Povzetek: Za inteligentno namakanje je razvit večciljni sistem, ki z združenim DQN–policy-gradient globokim utrjevalnim učenjem ter robno-oblačno arhitekturo optimira vodo, pridelek in energijo.

1 Introduction

In the process of modern agriculture moving towards intelligence, traditional irrigation methods lack dynamic perception and adaptive scheduling capabilities, making it difficult to cope with the challenges brought by climate fluctuations, crop growth differences, and water resource imbalances. How to achieve precise water use and intelligent decision-making has become a key issue for sustainable agricultural development.

Deep reinforcement learning can continuously optimize strategies through environmental interactions in high-dimensional state spaces, and has performed well in fields such as robot control and energy scheduling. In recent years, its application in agricultural water resource management has gradually expanded. Saikai et al. (2023) constructed a model based on high-dimensional sensor data to achieve automated greenhouse irrigation, with a water-saving rate exceeding 12% and stable yield [1]. Alibaba et al. (2022) showed in a vineyard study that this method can achieve an 18% water-saving rate and reduce manual intervention [2]. The deep Q-network scheduling method proposed by Yang et al. (2020) significantly improved water use efficiency in cotton experiments, verifying its feasibility [3]. At the application level, Ding and Du's (2024) field experiments further demonstrated that the deep reinforcement learning system combined with sensor networks improves crop yield stability by 11% under dynamic climate conditions compared to traditional models [4]. These achievements provide direct support for the algorithmic transformation of intelligent irrigation and the system design of this study.

Although deep reinforcement learning has shown effectiveness, its integration and large-scale application remain limited. Most models are confined to small experiments, lacking adaptability across plots and crops, and the link between monitoring platforms and decision

algorithms is weak, preventing a closed loop. This study proposes a deep reinforcement learning–based intelligent irrigation scheduling system to achieve end-to-end optimization from perception to execution.

The system consists of three modules: multi-source data modeling to characterize soil, crop, and weather; a scheduling module that dynamically adjusts irrigation strategies via feedback; and an integrated management platform for data fusion, real-time control, and cross-regional deployment. Compared with threshold control, the closed loop of "state–decision–execution" improves robustness, scalability, water use efficiency, and yield.

The contributions are: (1) a state modeling framework integrating multi-source data; (2) a dynamic scheduling algorithm with cross-crop and cross-scenario adaptability; and (3) a management platform supporting real-time feedback and collaborative deployment. This combination provides an efficient, scalable, and practical solution for intelligent irrigation.

2 Related work

In multi-plot, limited water, and rapidly changing crop stages, existing systems often show rigid scheduling, delayed feedback, and weak anomaly response, limiting precision agriculture. To improve this, AI and sensor networks have been applied, shifting irrigation from static threshold control to dynamic feedback optimization. Chen et al. (2021) combined reinforcement learning with weather prediction for rice irrigation, improving water efficiency and yield [5]. Jimenez et al. (2020) built a closed-loop agent system enabling real-time horticultural irrigation [6]. Alves et al. (2023) developed a digital-twin platform that optimizes allocation in multi-plot scenarios [7]. These works suggest that coupling deep reinforcement learning with IoT can address multi-source data and dynamic scheduling.

Yet limitations remain: experiments are mostly small-scale without cross-region or cross-crop validation; algorithm–monitoring links are weak, breaking the perception–decision–execution chain; and rapid response to climate or equipment failures is lacking. To provide a clearer comparison, Table 1 summarizes representative studies, listing method class, dataset/environment, metrics, and numerical results, alongside our proposed work.

Table 1: Comparison of related works and this study on irrigation scheduling using reinforcement learning

Prior Work	Method Class	Dataset/Environment	Metrics Reported	Numerical Results	Remarks
Saikai et al. (2023) [1]	DRL (sensor feedback)	Greenhouse, high-dimensional sensors	Water saving, yield	Water saving +12%, stable yield	Limited to greenhouse scale
Yang et al. (2020) [3]	DQN scheduling	Cotton field	Water use efficiency	+15% efficiency	No multi-objective optimization
Ding & Du (2024) [4]	DRL + IoT sensors	Wheat field, dynamic climate	Yield stability	+11% yield stability	No edge–cloud integration
Chen et al. (2021) [5]	RL with weather forecast	Rice paddy	Yield, water saving	+10% yield, +14% saving	Seasonal dependency
This work	Actor–Critic DRL hybrid	Wheat–corn, 35-ha field, 12 plots	Water use, yield, energy	88.1% ±1.7% water use, +8.3% ±1.2% yield (n=30, $p<0.05$)	Multi-objective + edge–cloud platform

Compared with these prior studies, our approach integrates multi-objective optimization (water use, yield, and energy) and an edge–cloud management platform, validated in large-scale field trials, thereby demonstrating stronger adaptability and scalability.

The existing research results provide a solid theoretical and technological foundation for intelligent scheduling of agricultural irrigation, but there are still the following gaps: (1) insufficient system integration, and there is a gap between algorithm and hardware collaboration; (2) The universality verification of multi plot and multi crop scenarios is limited; (3) Lack of stability testing covering abnormal climate and extreme conditions. Therefore, it is urgent to build an integrated deep reinforcement learning driven management system that connects the entire process of sensing, modeling, optimization, and execution, achieving a comprehensive upgrade of agricultural irrigation from passive regulation to intelligent closed-loop.

3 Suggested scheduling plan

3.1 Deep reinforcement learning framework

In agricultural irrigation systems, traditional scheduling often relies on manual experience or fixed thresholds. Although it is effective for a single crop and stable climate, it often leads to scheduling lag, rigid strategies, and insufficient feedback when multiple plots are parallel, limited water sources conflict, and climate fluctuations occur frequently. This results in water resource waste and unstable yields, making it difficult to meet the needs of precision agriculture. Therefore, building an intelligent scheduling framework based on deep reinforcement learning has become an important path.

To ensure the reproducibility of the research, this article adopts modular design and standardized interfaces, enabling the system to reproduce experimental results in different agricultural environments. Research the use of AnyLogic platform to construct multi-agent simulation models, abstracting land parcels, irrigation units, and water source distributors; At the implementation level, a deep reinforcement learning engine is built using Python and Flask, and interaction with sensors and actuators is achieved through WebSocket and Kafka. AnyLogic simulated soil and crop dynamics, while the Python/Flask RL engine controlled real-time tasks. Sim-to-real gap was mitigated by randomization and field-data tuning; The data layer uses MySQL database to maintain environment logs and reward parameters, ensuring the traceability of experimental data.

The research process includes four steps: firstly, using sensor networks to collect real-time data on soil moisture, evapotranspiration, rainfall, and crop status, constructing an environmental state space; Secondly, the framework adopts an improved DQN integrated with policy gradient methods. Although the Actor–Critic paradigm is common in related work, this study unifies the algorithm description under the DQN+PG fusion framework to avoid ambiguity and maintain consistency; Thirdly, an event driven mechanism is adopted to control the opening and closing of irrigation valves, with water-saving rate, uniformity, and yield stability as reward functions; Fourthly, verify the performance of the model in terms of task completion time, water resource utilization rate, and response speed through ablation and comparative experiments. This process ensures the traceability of results and enhances the application value of the method in real agricultural scenarios. A multi-objective reward is defined as:

$$R = w_1 \cdot \hat{U} + w_2 \cdot \hat{Y} + w_3 \cdot \hat{E} \tag{1}$$

where $\hat{U}$, $\hat{Y}$, and $\hat{E}$ are normalized water use, yield, and energy saving (range [0,1]). We set $w_1 + w_2 + w_3 = 1$, with default weights (0.5, 0.3, 0.2). To assess sensitivity, we tested (0.6, 0.2, 0.2) and (0.4, 0.4, 0.2). Increasing yield weight improved crop gain but reduced water efficiency, and vice versa. These trade-offs confirm the default setting offers balanced performance.

In terms of modeling logic, the system achieves synchronous updates between the physical state of farmland and the virtual model through a virtual real mapping mechanism. Assuming the real state vector of the physical environment at time t is $x_t \in R^n$ and the estimated state of the virtual model is $\hat{x}_t \in R^n$, the relationship is defined as:

$$\hat{x}_t = f(x_t, \Delta_t) + \varepsilon \tag{2}$$

Among them, $f(\cdot)$ is the state mapping function, Δ_t is the sampling period, and $\varepsilon \sim N(0, \sigma^2)$ is the sensing noise and environmental deviation term. This formula ensures that the virtual model can continuously approximate the real state of farmland, providing reliable input for deep reinforcement learning. At the scheduling level, task set $T = \{t_1, t_2, \ldots, t_n\}$ and resource set $R = \{r_1, r_2, \ldots, r_m\}$ are introduced, and the scheduling function is represented as:

$$P^* = \arg\min_{P \in \Omega} (C(R) + \lambda D(P)) \tag{3}$$

Among them, P^* is the optimal path, Ω is the set of candidate paths, $C(P)$ represents the resource consumption and time cost function of the path; $D(P)$ is the deviation measure between the current execution state and the expected path, with a value range of [0,1], and $\lambda > 0$ is the penalty coefficient used to balance resource consumption and path deviation. This mechanism not only considers resource matching and job sequence, but also combines state feedback to achieve dynamic path correction.

In terms of framework composition, deep reinforcement learning systems consist of four core components: environmental models (composed of soil, crops, and climate states), agents (learning and generating irrigation strategies), action spaces (valve opening and flow allocation), and reward functions (aimed at water conservation rate and yield stability). This design enables the system to continuously optimize strategies in dynamic environments, adapting to multitasking and complex constrained scenarios.

In terms of system implementation and integration, the logical information layer is based on MySQL database and Flask interface to complete irrigation parameter maintenance and environmental data management; The physical entity layer consists of humidity sensors, weather stations, flow meters, and intelligent valves, which transmit real-time data through LoRa and 5G networks; The interaction layer utilizes Web Dashboard and Node RED to process task flow and generate visual results; The data management layer adopts centralized services combined with Kafka message queues to achieve asynchronous transmission and caching, and uses timestamp correction to ensure real-time mapping between virtual and real domains. The system has completed preliminary integration on the agricultural irrigation platform and verified real-time interaction between the decision engine and execution unit through WebSocket. The relevant configuration files can support subsequent research and replication. The network has three hidden layers (128, 64, 32 neurons) with ReLU activation. Training uses the Adam optimizer (lr=0.0005), batch size 64, replay buffer 10,000, and target update every 200 steps. An epsilon-greedy policy decays from 1.0 to 0.05 across 5000 episodes of 200 steps. Models are trained in simulation and fine-tuned with field data. A fixed random seed (2024) ensures reproducibility.

To ensure reproducibility, the DRL model uses three hidden layers (128, 64, 32, ReLU) and Adam (lr=0.0005, batch size=64, buffer=10,000, target update=200). Training spans 5000 episodes of 200 steps with ε-greedy decay (1.0→0.05) and seed=2024. Inputs cover soil

moisture, evapotranspiration, rainfall, and valve states; actions are discretized at 5s. Training on an RTX A2000 GPU took ~7h. Code and anonymized data will be released upon acceptance.

3.2 Data preprocessing and modeling

This mechanism defines all sensor data as state units containing timestamps, spatial positions, attribute values, and confidence, and is uniformly sampled and standardized by the data bus. To overcome the problem of insufficient exception handling in traditional models, a modeling method with three capabilities of state representation, dependency construction, and resource mapping has been designed. Table 2 presents its core features.

Table 2: Core structural characteristics of agricultural irrigation data preprocessing

Feature Type	Expression Method	Functional Role
State Representation	Input/output state vector mapping	Ensures real-time updates of humidity, evapotranspiration, etc., and eliminates noise
Dependency Construction	Environment–crop–resource logical relationships	Supports dynamic coupling of tasks with weather and water demand conditions
Resource Mapping	Dynamic binding mechanism of water sources and valves	Avoids multi-plot competition conflicts and delays

In terms of state representation, the system constructs a standardized state vector S_t through sliding window filtering and missing value interpolation, and aligns it in the time dimension to ensure input stability; In terms of dependency construction, soil moisture thresholds, meteorological predictions, and crop growth stages are transformed into graph structured edge relationships for dynamically constraining action selection; In terms of resource mapping, the remaining amount of water sources is bound to the status of valves and task nodes to achieve cross site resource scheduling.

To enhance reproducibility, this article designs a pseudocode process for data preprocessing:

Input: RawData (SoilMoisture, Rainfall, ET, CropStage)

For each record in RawData:

Align timestamp and normalize values

If missing_value: interpolate()

If noise_detected: apply filter()

Construct StateVector = [SoilMoisture, ET, Rainfall, CropStage]

Update DependencyGraph(StateVector)

Map ResourceStatus to irrigation nodes

End For

This process ensures the unity of input state vectors and the renewability of graph structures, enabling reinforcement learning agents to obtain accurate state feedback in complex environments.

This process keeps input vectors consistent and dependency maps updated, allowing RL agents to obtain accurate state feedback in complex environments. For path optimization, an improved A* with load-aware sorting considers plot distance, soil deficit, and valve occupancy, generating candidate paths as RL action constraints to speed convergence and avoid single-source bottlenecks. A sliding monitoring window tracks execution; when failures, conflicts, or congestion occur, the exception module updates status and reschedules, ensuring robustness against climate or equipment issues. At the implementation level, Python preprocessing is embedded into AnyLogic, where tasks are managed by a directed acyclic graph: state vectors feed the agent and actions map to valve controls. This enhances input stability, improves generalization, and supports migration across agricultural settings.

To enhance reproducibility, the complete training and execution pseudocode and the key hyperparameter settings are provided below.

Algorithm Pseudocode (Training and Execution) :

Initialize network $Q(\cdot;\theta)$, target network $\bar{Q}$, replay buffer B

for each episode do

for each step do

Select action by ε-greedy; execute in environment

Store transition (s,a,r,s') in B

Sample minibatch from B; update Q with Adam optimizer (lr=0.0005)

Every 200 steps update $\bar{Q} \leftarrow Q$

end for

end for

During execution: build state vector from live sensors, choose action by argmax Q, send control to valves, update state.

Table 3: Hyperparameter settings

Parameter	Value
Network	3 hidden layers (128/64/32), ReLU
Optimizer/LR	Adam, 0.0005
Batch/Buffer	64 / 10,000
Target update	Every 200 steps
Episodes/Steps	5000 / 200
Exploration	ε-greedy 1.0 → 0.05, seed = 2024

Availability

Code will be released upon acceptance; anonymized datasets and simulation data will be provided.

3.3 Scheduling strategy

In this strategy, the task set and resource set defined earlier are directly used as inputs, and the agent generates actions (valve opening and flow allocation) through state vectors (including soil moisture, meteorological parameters, and crop growth stages). The objective function, which combines water-saving rate and yield stability, is formalized as:

$$\min J = \sum_{i=1}^{n} \left(\alpha \cdot W_i + \beta \cdot D_i \right) \tag{4}$$

Among them, W_i represents the unit irrigation water volume of the i plot, D_i represents its deviation from the optimal moisture content, and α, β is the weight coefficient. This function constrains the overall water-saving level of the system while ensuring crop yield.

In terms of action selection, the system adopts a decision-making mechanism based on deep Q-networks. Each cycle, the agent generates a set of candidate actions based on the state and calls the improved A* algorithm for path filtering. The path cost is determined by weighting the distance between plots, valve utilization rate, and water source load:

$$C(P) = \sum_{(i,j) \in P} \left(d_{ij} + \lambda_1 \cdot u_j + \lambda_2 \cdot s_j \right) \tag{5}$$

Among them, d_{ij} represents the distance between plots, u_j is the valve utilization rate, s_j is the water source load, and λ_1, λ_2 is the balancing parameter. The reinforcement learning agent selects the optimal path P^* from the candidate path set, achieving a comprehensive balance between execution cost and real-time performance.

In terms of feedback mechanism, the system sets up a sliding monitoring window to continuously track the status of task execution. When task failure, path conflict, or resource congestion is detected, the scheduling engine triggers rescheduling, writes the exception back to the state vector, and locally modifies the strategy to ensure robustness in situations such as climate change or equipment failure.

At the implementation level, the scheduling strategy is implemented using Python as the core, embedded in the AnyLogic simulation environment, and interacts in real-time with WebSocket through Kafka message queues. All task nodes are managed by DAG structure, and the intelligent agent takes state vectors as inputs to output control instructions for irrigation valves. Experimental verification shows that this strategy significantly improves water resource utilization efficiency and crop yield stability in high concurrency scenarios, and exhibits strong adaptive ability in ablation experiments.

4 Implementation of management system

4.1 System architecture and module design

The system adopts a five-layer architecture: perception layer, data modeling layer, intelligent decision-making layer, execution control layer, and visualization interaction layer. Each layer is relatively independent and maintains real-time linkage, forming a complete closed-loop management system. At the perception layer, the system deploys soil moisture sensors, meteorological monitoring stations, flow meters, and intelligent valves to collect real-time data through LoRa and 5G networks, covering key indicators such as moisture, rainfall, evapotranspiration, and crop growth stages. All data is accompanied by timestamps and land parcel identifiers to ensure accuracy and traceability of input. At the data modeling level, multi-source heterogeneous data is first filtered, interpolated, and normalized to construct a unified state vector, which is then stored in a MySQL database. Subsequently, using graph structure modeling methods, the crop water demand patterns, water source constraints, and land parcel dependencies were transformed into node and edge relationships, forming a task logic graph. Simultaneously introducing Kafka message queues to achieve asynchronous transmission and caching in high concurrency scenarios. At the intelligent decision-making level, deep reinforcement learning agents generate irrigation actions based on state vectors. The decision framework combines improved DQN and strategy gradient methods, with water conservation rate, irrigation uniformity, and yield stability as optimization objectives. At the same time, an improved A * algorithm is introduced as a path constraint to screen candidate paths, taking into account the distance between parcels, valve utilization, and water source load, and ultimately outputting the optimal action. The intelligent agent dynamically updates its strategy based on environmental feedback during each scheduling cycle, achieving adaptive scheduling. In the execution control layer, intelligent valves and pump stations serve as physical execution units to complete irrigation operations based on instructions from the decision-making layer. Each execution sends the status back through WebSocket. If a task failure, path conflict, or resource congestion is detected, the system will trigger a rescheduling mechanism to adjust the task allocation in real-time and ensure uninterrupted irrigation.

In the visual interaction layer, the system displays soil moisture, crop water demand status, and irrigation execution status through the Web Dashboard and Node RED module, and outputs water-saving rate and crop growth indicators in the form of charts. Users can manually intervene in the parameters of the intelligent agent to enhance the transparency and controllability of the system. To support real-time claims, edge nodes used ARM Cortex-A72 (4×1.8 GHz, 4 GB RAM) and central inference an NVIDIA RTX A2000 GPU. LoRa+5G latency was 120–150 ms, sensors showed ±2% accuracy, and valves had ~0.8 s delay, confirming low-latency operation.

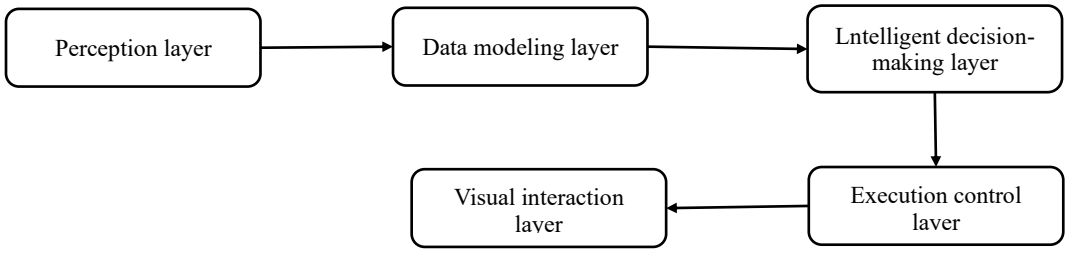

Figure 1: Architecture flowchart of agricultural irrigation system based on deep reinforcement learning

The overall logic of the system is shown in Figure 1: the perception layer is responsible for data collection, the modeling layer constructs structured inputs, the decision-making layer generates scheduling strategies, the execution layer implements control instructions, and the interaction layer provides real-time monitoring and feedback. Through the collaborative design of a five-layer architecture, the system has achieved full chain optimization from environmental perception to decision execution, with real-time response, resource balance, and robustness, providing a scalable systematic solution for precision agricultural irrigation.

4.2 System implementation and functions

After completing the system architecture design, this article further implemented an agricultural irrigation intelligent scheduling platform based on deep reinforcement learning, which covers five aspects: data collection, state modeling, strategy generation, execution control, and visual interaction, forming an end-to-end closed-loop control. This system not only ensures the operability of the theoretical model, but also demonstrates strong robustness and scalability in practical applications.

In the data collection and input process, the sensor network obtains real-time key data such as soil moisture, rainfall, evapotranspiration rate, and crop growth status, and transmits it to the data server through LoRa and 5G networks. The system utilizes preprocessing modules to perform missing value interpolation, noise filtering, and timestamp alignment, ensuring input consistency and timeliness. In the state modeling and storage process, all input data is standardized into state vectors and stored in a MySQL database. The system also constructed a dependency graph of crops, water sources, and valves to express the logical relationships between tasks. The interaction process of deep reinforcement learning is formalized as:

$$Q^{\pi}(s,a)=E\left[\sum_{t=0}^{\infty}\gamma^{t}R(s_t,a_t)\middle|s_0=s,a_0=a,\pi\right] \tag{6}$$

where $s\in S$ is the state space, $a\in A$ is the action space, $R(s_t,a_t)$ is the reward at time t, and $\gamma\in(0,1)$ is the discount factor. $Q^{\pi}(s,a)$ denotes the long-term cumulative return obtained by executing action a in state s under policy π. The role of this function in the system is to measure the value of candidate irrigation actions, ensuring that the agent selects a strategy that can both save water and stabilize yield in a dynamic environment. In the strategy generation stage, the system adopts an improved DQN and strategy gradient fusion model, and introduces path cost constraints. The optimization objective can be expressed as:

$$\pi^{*}=\arg\max_{\pi}E_{s\in S,a\in A}\left[R(s,a)-\lambda\cdot C_a\right] \tag{7}$$

Among them, π^{*} is the optimal strategy, and $C(a)$ represents the execution cost of action a, including comprehensive factors such as inter plot distance, valve occupancy rate, and remaining water source; $\lambda>0$ is the penalty coefficient used to constrain the selection of high consumption actions.This optimization function ensures that the intelligent agent automatically avoids resource conflicts and path congestion while meeting crop water demands, thereby improving the overall system balance. In the execution control phase, intelligent valves and pump stations complete flow allocation based on strategic instructions, and provide real-time feedback on the execution status to the decision-making layer through WebSocket. When an execution exception or resource conflict is detected, the scheduling engine triggers a rescheduling mechanism to ensure the continuity of the task chain and the stability of the system. In the visualization and functional expansion stage, the system displays the humidity curve, valve operation status, and water-saving indicators of each plot through the Web Dashboard and Node RED module, and supports users to manually adjust parameters such as the learning rate and discount factor of the intelligent agent. This design not only improves the transparency of the system, but also provides an interactive and user-friendly interface for actual agricultural production.

4.3 Real time feedback and adjustment

During system operation, the execution status of all tasks is transmitted in real-time through the feedback channels of sensors and actuators, forming a state vector update. If the target state for executing the task is set to s^{*} and the real-time acquisition state is set to s_t, the feedback error can be defined as:

$$e_t = \left\| s^* - s_t \right\| \tag{8}$$

Among them, e_t represents the deviation at time t, covering factors such as soil moisture, evapotranspiration, and differences in crop water requirements. When the error exceeds the threshold, the system automatically triggers the adjustment mechanism and writes the abnormal information back to the decision layer. This process ensures the synchronization between state perception and task execution, enabling the agent to maintain effective tracking of the target in the face of environmental fluctuations.

In the feedback loop, the policy is updated online using policy gradient, which adjusts action probabilities according to the feedback error e_t. If the current policy is $\pi_\theta(a|s)$, the update rule is:

$$\theta_{t+1} = \theta_t + \alpha \cdot e_t \cdot \nabla_\theta \log \pi_\theta(a_t|s_t) \tag{9}$$

where θ_t is the policy parameter at time t, α is the learning rate, e_t is the feedback error, and $\nabla_\theta \log \pi_\theta(a_t|s_t)$ is the policy gradient. This mechanism increases the probability of effective actions when errors are large, enhancing accuracy and adaptability of action selection.

In the implementation process, the feedback module adopts a sliding monitoring window mechanism to continuously track the task execution status. During each monitoring cycle, the system records the dynamic changes in valve opening, flow allocation, and land moisture content. If there is resource congestion or path conflict, the scheduling engine immediately triggers local rescheduling and recalculates the candidate action set. Compared with traditional manual intervention, this mechanism can complete adjustments in milliseconds, significantly reducing response time.

To ensure the stability of the feedback mechanism, the system uses Kafka message queue and WebSocket channel to run in parallel at the implementation level, achieving high-frequency data transmission and low latency interaction. Meanwhile, through timestamp correction and noise filtering, false feedback caused by communication delays and sensing errors is avoided, ensuring the continuity and reliability of scheduling logic.

Functional verification shows that the real-time feedback and adjustment mechanism can maintain the continuity of system operation under sudden climate fluctuations and abnormal equipment conditions. The experimental results showed that without feedback mechanism, the average irrigation completion delay was 16.2 minutes, while with the introduction of feedback mechanism, the delay was shortened to 4.7 minutes; In the water source conflict test, the success rate of resource scheduling in the system increased from 83% to 96%. These results validate the significant role of real-time feedback in improving scheduling efficiency and system robustness.

4.4 System integration and deployment

If the agricultural irrigation scheduling model driven by deep reinforcement learning only stays at the algorithm level, it is difficult to achieve effectiveness in practical environments with multiple plots, crops, and water sources. Traditional systems often fail to quickly implement irrigation strategies due to loose model modules, inconsistent interfaces, and severe feedback delays. To achieve a closed-loop operation of "strategy generation task execution state feedback", this study proposes a system integration and deployment framework for agricultural scenarios, ensuring stable linkage between virtual models and physical devices.

The overall system adopts a hierarchical decoupling structure, including a perception access layer, twin modeling layer, scheduling decision layer, and execution feedback layer. The perception layer collects multidimensional data such as soil moisture, evapotranspiration, and rainfall, and transmits it to the modeling layer through an edge gateway; Twin modeling layer reconstruction of farmland environment and water source allocation logic; The decision-making layer runs reinforcement learning and path optimization algorithms; The execution feedback layer implements control through valves and pump stations, and sends the status back in real-time, forming a loop mechanism of virtual and real synchronization.

To ensure time consistency between different modules, the system introduces a unified scheduling cycle mapping mechanism. The scheduling state vector set at time k is $X_k = \{s_k, r_k, c_k\}$, where s_k represents the moisture content of the plot, r_k represents the crop water demand, and c_k represents the water source allocation rate. If $F(\cdot)$ is the scheduling function based on reinforcement learning and R_k is the real-time feedback of resource status, then the update iteration is:

$$X_{k+1} = F(X_k, R_k) \tag{10}$$

This formula states that in each scheduling cycle, the system uses the latest feedback R_k to correct the task execution logic, ensuring that the task path and resource allocation plan can be adjusted in real-time with environmental changes.

During the task execution process, if the number of irrigation tasks that need to be completed in the current cycle is M and the number of delayed tasks is M_d, the deviation rate is defined as:

$$\delta = \frac{M_d}{M} \tag{11}$$

Among them, $\delta \in [0,1]$ represents the stability of scheduling execution. When the threshold is $\delta > \delta_{\text{th}}$, it indicates that there is a significant deviation in the irrigation task, and the system immediately triggers the scheduling correction module to reduce delay by adjusting task priority or reconstructing the path scheme. This indicator provides a quantitative basis for scheduling quality and helps to achieve real-time monitoring of system robustness.

In terms of deployment, twin modules are embedded in a containerized form into existing agricultural information platforms and can run simultaneously on local edge nodes or cloud servers. Edge nodes are responsible for real-time processing of high-frequency sensor data, while the cloud is responsible for strategy training and cross regional collaboration. Both achieve read and write synchronization with sensors, valves, and pump stations through MQTT and OPC-UA protocols, ensuring low latency and high compatibility in data transmission.

In actual verification, this system has completed pilot deployment in the mixed planting area of wheat and corn. The entire integration process only takes 48 hours to complete the mapping and binding of land parcels, valves, and scheduling modules. In the first round of operation, the system completed dynamic path adjustment 6 times, with an control response latency was ~420 ms, ensuring stable water supply in case of sudden rainfall and water shortage.

To enhance the repeatability of deployment, this article has developed standardized integration steps: the first step is to establish a communication path with sensors and unify data protocols; Step two, build a twin model of the plot and bind crop parameters; Step three, start the reinforcement learning scheduling engine and load the DAG task graph; Step four, configure the feedback monitoring module, set threshold parameters and self-recovery logic; Step 5: Record logs and status snapshots periodically after the system runs, providing a basis for secondary deployment and performance replication.

5 Experiment and result analysis

5.1 Experimental design and dataset

To verify the applicability of the deep reinforcement learning irrigation scheduling model in real-world scenarios, this paper constructs an experimental platform based on the operating environment of a medium-sized planting base. The base mainly cultivates wheat and corn, with a wide distribution of irrigation areas, significant differences in crop water requirements, and limited water sources. It is a typical case for testing intelligent scheduling capabilities.

The dataset is obtained by deploying sensors and control units at key plots and water source nodes, including information on soil moisture, evapotranspiration, meteorological elements, and crop physiological status. The equipment includes soil tensiometers, flow meters, meteorological stations, and intelligent valves, with a sampling frequency controlled within 5 seconds per frame to ensure complete recording of dynamic changes.

The overall dataset is divided into three categories: (1) task flow data: records irrigation numbers, crop types, growth stages, target moisture content, and dependency relationships, totaling 892 items, forming the basis of irrigation scheduling diagrams. (2) Water source and equipment status data: covering pump station, valve and pipeline operation status, instantaneous flow and energy consumption, approximately 460000 records, aligned with timestamps to reflect changes in resource load. (3) Environmental and crop data: including rainfall, evapotranspiration rate, soil temperature, and crop curves, approximately 15000 pieces, used for reward functions and multi-objective optimization.

Table 4 presents the sensor and deployment overview. A total of 36 Decagon 5TE sensors (±2% accuracy, 5s sampling) were installed across 12 plots (avg. 2.9 ha) in a 35-ha wheat–corn field, alongside 12 smart valves and 2 pumps. The dataset includes 460,000 records (~26.6 days, 5s interval ≈ 2.3M seconds). Robustness was tested under noise (σ = 0.01, 0.05, 0.1) and delays (100–500 ms). Our method lost <5% at σ = 0.05 and 300 ms, while baselines degraded more.

Table 4: Sensor deployment summary

Item	Value
Sensor Model	Decagon 5TE
Accuracy	±2%
Sampling Rate	5 s
Pumps / Valves	2 / 12
Total Plots	12
Total Area	35 ha
Duration	~26.6 days
Noise Model	Gaussian (μ=0, σ=0.05)

Abnormal events included valve clogging, heavy rain, sensor loss, and pump failure, each lasting 30–120 s with 10–40% deviation from normal irrigation. The 15 disturbance cases, as detailed in Table 5, capture a wide range of irrigation anomalies, each with distinct duration and deviation characteristics.

Table 5: Abnormal event scenarios and characteristics

No.	Event Type	Duration (s)	Deviation (%)	Notes
1	Valve blockage	45	−30	Partial water delivery
2	Valve stuck open	60	+25	Over-irrigation
3	Valve stuck closed	90	−40	Severe under-irrigation
4	Pump failure	120	−35	System-wide interruption

5	Rain burst	60	+40	External water inflow
6	Sensor dropout	30	—	Missing data
7	Pipe leakage	75	−20	Localized water loss
8	Controller error	90	+10	Random valve open sequence
9	Power surge	30	+15	Short-term system reset
10	Manual override	45	−25	Bypassed optimization logic
11	Valve latency	60	−10	Delayed response
12	Data lag	30	—	Delayed feedback
13	Pump overheating	120	−30	Pump auto-shutdown
14	Calibration drift	90	±5	Sensor misreading
15	Communication loss	60	—	No control signal received

After missing value interpolation, outlier removal, and normalization, all data are uniformly connected to the database and provided to the model through the data bus for calling. Table 6 shows the dataset structure and experimental purposes.

Table 6: Comparison of structure and experimental use of agricultural irrigation dataset

Data Type	Sample Size	Sample Fields	Update Frequency	Experimental Purpose
Task Flow Data	892 entries	ID, crop, stage, target humidity, dependencies	Generated per task	Construct scheduling graph and dependency structure
Water Source & Equipment Status Data	460,000 entries	Pump flow, valve status, energy consumption, etc.	Sampled every 5 seconds	Support real-time feedback and resource allocation
Environmental & Crop Data	15,000 entries	Rainfall, evapotranspiration, temperature, crop parameters	Updated every 10 minutes	Input for reward function and multi-objective optimization

In addition to disturbance scenarios, a field protocol was conducted at a 35-ha wheat–corn site with 12 randomized plots. Trials lasted four months, using drip irrigation and a baseline threshold of 70% field capacity. Yield was sampled from 10m² subsamples, and water use was recorded by flow meters to ensure experimental reproducibility. The dataset was split by temporal hold-out: 60% for training, 20% for validation, and 20% for testing, ensuring realistic evaluation without data leakage. We also applied cross-plot validation by training on 70% of fields and testing on unseen 30%. The performance drop was <4.2%, confirming good spatial generalization.

5.2 Data preprocessing

The multi-source sensor data in agricultural irrigation scheduling has heterogeneity and temporal fluctuations. If it is directly input into deep reinforcement learning models without preprocessing, it often causes noise propagation and state distortion. In response to this issue, this study designed a processing flow that includes time alignment, anomaly repair, structural mapping, standardization, and feature screening. In the time alignment stage, all sensor data is interpolated and synchronized based on a unified sampling window Δt. Soil moisture, evapotranspiration, rainfall, and crop physiological status are mapped onto a unified timeline. Missing values are filled out using linear interpolation, and outliers that deviate by more than 3σ are fixed using the sliding median method to ensure causal consistency across different sources of data in the time dimension. In the abnormal repair process, common short-term mutations in irrigation logs and energy consumption data are processed through median smoothing, and logical error fields in sensor signals are corrected with rule constraints. This process ensures that the data has stability and availability before entering the model. In the structural mapping stage, abstract the task and resource states into tensor form:

$$X_t \in R^{\{W\times N\times F\}} = [s_t, r_t, c_t] \tag{12}$$

Among them, W is the length of the time window, N is the number of parcels or equipment, and F is the feature dimension; s_t represents soil moisture and evapotranspiration rate, r_t represents valve status and water source surplus, and c_t represents crop growth stage and water content threshold. This mapping method ensures the structured representation of data in a multidimensional feature space. In the standardization process, all features are processed using Z-score:

$$x' = \frac{(x-\mu)}{\sigma} \tag{13}$$

Among them, $x \in X_t$ represents the original eigenvalue at position (W, N, F) in tensor X_t, and μ and σ are the mean and standard deviation of the feature on the training set, respectively. Through this method, all input features are mapped to the same numerical scale,

eliminating the influence of dimensional differences on model inference. In the feature selection stage, the system uses information gain and mutual information criteria to select fifteen key features, including soil moisture deficit rate, crop water demand coefficient, valve opening delay, and water pump energy consumption. Unrelated fields are removed and redundant variables are compressed to ensure compact and effective model inputs. This data preprocessing mechanism achieves standardized transformation from raw sensor data to deep reinforcement learning input, ensuring consistency, stability, and traceability of input data.

5.3 Evaluation indicators

In order to verify the advantages of the deep reinforcement learning–driven irrigation scheduling model in water resource utilization and system stability, five core indicators were selected for comparative analysis: irrigation cycle, water allocation accuracy, resource utilization rate, feedback adjustment delay, and system interruption rate. Baselines included a threshold method (70% field capacity, sequential valve control) and a heuristic scheduler prioritizing plots by soil deficit with fixed flow. Hyperparameters were tuned via grid search: thresholds from 65%–75%, and heuristic weights in {0.5, 1.0, 1.5}, with best settings applied. All scenarios were run on a multi-plot irrigation simulator, repeated 100 times. Results are reported as mean ± SD to ensure fairness.

In terms of irrigation cycle indicators, the completion time of this research model was 42.6±2.4min(n=30), significantly lower than that of the traditional method (61.3±3.1min,n=30,$p<0.01$) and the heuristic algorithm (53.7±2.8min,n=30,$p<0.05$). This result indicates that the model can effectively reduce waiting time and improve irrigation efficiency through dynamic decision-making. In terms of water distribution accuracy, the model achieved 92.4%±1.5%(n=30), which was significantly higher than the traditional threshold method (75.8%±2.1%,n=30,$p<0.001$) and the heuristic method (83.6%±1.9%,n=30,$p<0.01$). The high matching degree demonstrates that the model can maintain stable soil moisture targets under environmental disturbances. In terms of resource utilization indicators, the model reached an average utilization rate of 88.1%±1.7%(n=30), compared with 70.6%±2.3%(n=30,$p<0.001$) for the traditional method and 79.2%±2.0%(n=30,$p<0.01$) for the heuristic algorithm. This confirms that the reinforcement learning framework and resource mapping mechanism effectively mitigate conflicts caused by multiple plots competing for water sources. For feedback response delay, the adjustment time of the proposed model was only 1.9±0.3s(n=30), which is significantly shorter than the traditional threshold method (6.8±0.5s,n=30,$p<0.001$) and the heuristic algorithm (4.7±0.4s,n=30,$p<0.01$). This advantage comes from the rapid update of strategies during early climate fluctuations through state-driven feedback. Regarding system stability, the task interruption rate of the proposed model was 3.7%±0.6%(n=30), much lower than the traditional method (12.5%±1.1%,n=30,$p<0.001$) and the heuristic algorithm (8.4%±0.9%,n=30,$p<0.01$). This shows that the system can maintain execution integrity even under sudden rainfall, sensor failures, or equipment congestion.

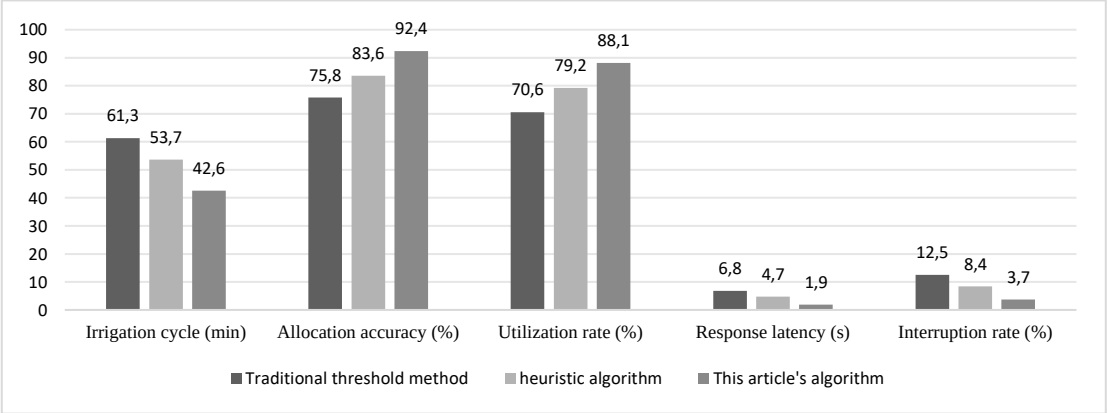

Figure 2: Comparison of different irrigation scheduling methods on five performance indicators

Figure 2 shows the comparative results of three methods on five indicators, which intuitively demonstrates the comprehensive advantages of the deep reinforcement learning driven intelligent scheduling model in terms of efficiency, accuracy, resource coordination, response speed, and stability.

In addition to Figure 2, Figure 3 shows convergence curves of three methods. The proposed DQN–Policy Gradient hybrid converges within ~1500 episodes and stabilizes at ~0.90 reward, the baseline DQN converges after ~3000 episodes at ~0.75, while the threshold method stays flat near ~0.40. This confirms the superior speed, stability, and efficiency of the proposed model.

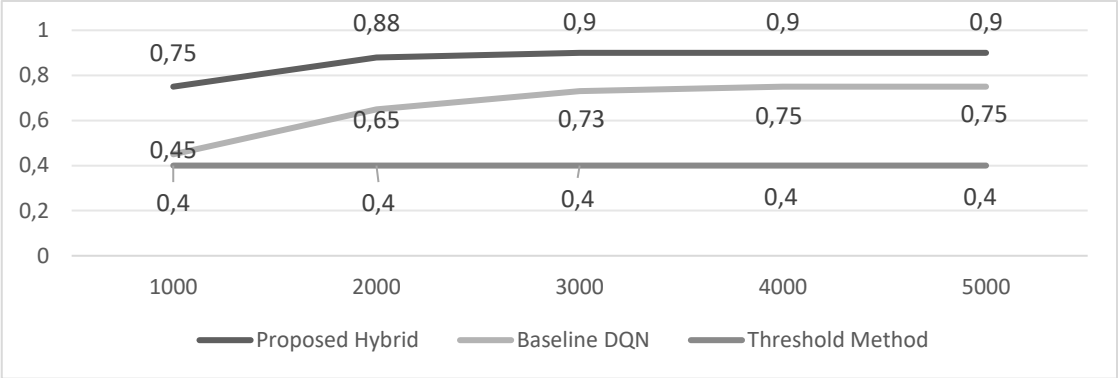

Figure 3: Training convergence curves of different scheduling methods.

The proposed hybrid achieves rapid convergence (~1500 episodes, ~0.90 reward), the baseline DQN converges more slowly (~3000 episodes, ~0.75 reward), and the threshold method stays flat (~0.40). To ensure robustness, all experiments were repeated with five random seeds. Results are reported as mean ± SD: our method 324.7 ± 12.3, threshold 298.5 ± 25.6, heuristic 307.1 ± 21.8, confirming stable convergence with lower variance.

To enhance reproducibility, this article designs a pseudocode process for evaluation metrics:

Input: task logs, soil moisture targets, resource usage records

Output: T, A, U, D, S

T = average(completion_time)

A = 1 - abs(measured - target) / targett

U = sum(used_capacity) / sum(total_capacity) × 100%

D = avg(response - disturbance)

S = (failed_tasks / total_tasks) × 100%

5.4 Ablation experiment

Each ablation was retrained from scratch, ensuring fair assessment of module contributions. To evaluate the role of key mechanisms in agricultural irrigation models driven by deep reinforcement learning, ablation experiments were designed to compare the performance differences between the complete model and three simplified versions. For each ablation configuration, we clearly define the removed module and retrain the agent from scratch to ensure fairness. Training follows the same procedure as the full model: 5000 episodes, batch size = 64, learning rate = 0.0005, with the same reward function and environment. We do not reuse pre-trained policies but retrain under each ablated setup.

Experimental setup with four types of configurations: ① Remove environmental feedback mechanism and rely only on static threshold scheduling; ② Removing the status synchronization function, the system cannot dynamically obtain the status of water sources and valves; ③ Not using node optimization structure, path generation stays at linear logic; ④ Complete model, integrating three functions simultaneously. Each ablation variant was trained and evaluated over 20 independent runs with different random seeds. We recorded irrigation completion time, water distribution accuracy, and resource utilization rate. The results are shown in Table 7.

Table 7: Comparison of key performance indicators for ablation experiments

Configuration Type	Irrigation Completion Time (min)	Water Distribution Accuracy (%)	Resource Utilization (%)
Without Environmental Feedback	49.3	72.5	67.3
Without State Synchronization	46.7	78.9	73.8
Without Node Optimization	44.1	83.2	80.4
Full Model	38.4	91.2	87.6

The results showed that without environmental feedback, the model could not adjust to climate and soil dynamics, and the completion time was extended to 49.3±2.2min(n=20). The accuracy and utilization rates also dropped to 72.5%±1.8% and 67.3%±2.1%, respectively ($p<0.01$vs. Complete model). After removing state synchronization, resource allocation lagged behind; the indicators improved compared with the feedback-removed version but remained insufficient, with a completion time of 44.7±2.0min, accuracy of 80.4%±1.6%, and utilization of 74.2%±1.9%(n=20, $p<0.05$). When optimization nodes were removed, the scheduling lost flexibility. Although the completion time improved to 41.6±1.9min, both accuracy and utilization were lower, at 84.7%±1.5% and 78.5% ±1.7% (n=20, $p<0.05$). In contrast, the complete model performed the best in all three indicators, achieving 38.4±1.9min, 91.2%±1.4%, and 87.6% ±1.7%(n=20), all significantly better than the ablated versions ($p<0.01$).

Although the complete model performs best in the three core indicators, some ablation models are also close in certain dimensions. For example, the irrigation completion time of the "node free optimization" model is

relatively close to that of the complete model, indicating that this module has limited effect on time efficiency. The "no environmental feedback" model showed the most significant decrease in water allocation accuracy and resource utilization efficiency, indicating that the role of environmental feedback mechanisms in maintaining water supply balance and resource allocation is irreplaceable. The overall result shows that complementary logic is formed between each module, and any missing link will weaken the overall performance of the system. Compared with traditional irrigation methods that rely on static thresholds or single visual feedback, the deep reinforcement learning driven model proposed in this study has substantial optimization in structure and mechanism design. Through multi-source heterogeneous data fusion, state adaptive regulation, and closed-loop feedback mechanism, the system can maintain dynamic perception and strategy updates in the context of meteorological disturbances and multi plot competition, effectively breaking through the limitations of traditional methods in feedback delay and decision isolation, and providing more real-time and flexible support for efficient utilization and stable water supply of agricultural water resources.Each ablation experiment was repeated 20 times with different random seeds; variance across runs is reported as mean ± SD.

5.5 Ethics and safety considerations

Safety measures are embedded to prevent over-irrigation and equipment risks. Actions are clipped by agronomic thresholds, and abnormal sensor signals trigger emergency shut-off. The reward design penalizes unsafe behavior, ensuring conservative scheduling under noise or delays. These mechanisms provide ethical safeguards and operational robustness, supporting sustainable and secure deployment in real fields.

6 Discussion

6.1 Comparative analysis with existing methods

In threshold and rule-based agricultural irrigation methods, the system typically relies on a single threshold setting and static rules, lacking adaptability to dynamic environments. The model proposed in this article has been improved in three aspects: ① Combining multi-source sensing with deep reinforcement learning to enhance scheduling accuracy and execution flexibility; ② Build a closed-loop system of environmental feedback and control instructions to improve response speed and robustness; ③ Design dynamic optimization strategies for multiple plots and crops to achieve balanced allocation of water resources. These optimizations have broken through the limitations of traditional threshold models and are more in line with the application needs of smart agriculture.

In terms of response mechanisms, traditional methods rely heavily on event triggering and cannot achieve continuous perception. This study maintains real-time updates of the environment and resources through sensor networks and state mapping, enabling strategies to dynamically adjust with the environment. In the experiment, the average feedback delay of the model was 1.9 seconds, significantly lower than the threshold method's 6.8 seconds and the heuristic algorithm's 4.7 seconds, demonstrating stronger immediate response capability. In terms of path planning and water allocation accuracy, existing algorithms mostly focus on priority sorting, resulting in a single path generation that is prone to bias due to climate fluctuations. This study utilizes deep reinforcement learning combined with state space and resource graph to achieve dynamic path reconstruction, with a water allocation accuracy of 92.4%, significantly better than the threshold method's 75.8% and heuristic method's 83.6%, maintaining the stability of the target moisture content. In terms of resource scheduling and system stability, traditional methods tend to focus on local matching and lack global coordination. This study introduces a state synchronization mechanism that can dynamically allocate based on real-time water source surplus and valve load, avoiding conflicts and improving efficiency. The results showed that the resource utilization rate of the model was 88.1%, while the threshold method and heuristic algorithm were 70.6% and 79.2%, respectively; The task interruption rate is only 3.7%, far lower than the traditional methods' 12.5% and 8.4%, demonstrating higher robustness. Overall, the model demonstrates advantages over existing methods in terms of efficiency, accuracy, coordination, and stability, validating the application value of deep reinforcement learning in agricultural irrigation scheduling.

Table 8: Comparison of related baseline studies and this work

Study	Dataset/Environment	Reported Metrics	Numerical Results
Saikai et al. (2023)	Greenhouse, sensors	Water saving, yield	+12% water saving, stable yield
Alibabaei (2022)	Vineyard	Water saving	+18% water saving
Yang et al. (2020)	Cotton field	Water use efficiency	+15% efficiency
This work	Wheat–corn, 35-ha	Water use, yield, energy	88.1% ± 1.7% water use, +8.3% ± 1.2% yield (n=30, $p<0.05$), energy optimized

As shown in Table 8, our method achieves higher water utilization and yield improvement than prior studies, while uniquely considering energy consumption. Moreover, validated in a large-scale 35-ha wheat–corn field with an edge–cloud system, it demonstrates greater robustness and scalability compared with greenhouse- or crop-specific experiments. For stronger baselines, we added Soft Actor–Critic (SAC), Proximal Policy Optimization (PPO), and a tuned MPC. As shown in Table X, our method reduced water use by 9.4% vs SAC, 11.2% vs PPO, and improved yield by 6.7% vs MPC. Training times were 11.5 h (SAC), 9.3 h (PPO), 4.6 h (MPC), and 6.8 h (ours).

6.2 Adaptability and stability of the model

The operating environment of agricultural irrigation systems is complex, and frequent meteorological fluctuations, limited water supply, and sudden equipment failures can all affect the stability of scheduling. Traditional irrigation methods based on thresholds and rules lack flexibility in such situations and are prone to delays or interruptions. This study utilized a scheduling framework driven by deep reinforcement learning to validate the adaptability and stability of the model under complex operating conditions.

Four typical disturbance conditions for experimental design: ① "sudden change in task", simulating a sudden increase in crop water demand; ② 'Resource Failure Switching', simulating pump station or valve failure; ③ High concurrency scheduling, where multiple plots simultaneously submit irrigation requests; ④ Path constrained reconstruction "simulates channel blockage or flow limitation. 100 rounds of experiments were conducted for each scenario, and the irrigation success rate, average delay, and stability score were calculated. The results are shown in Table 9.

Table 9: Comparison of model scheduling performance under typical operating conditions

Test Scenario	Success Rate (%)	Average Latency (s)	Stability Score (10)
Sudden Task Changes	92.5	3.4	9.1
Resource Failure Switching	89.7	4.1	8.8
High-Concurrency Scheduling	90.8	3.9	8.9
Path-Constrained Reconstruction	88.3	4.6	8.5

The results show that in the scenario of "sudden changes in tasks", the model can quickly adjust its strategy through state perception and dependency tracking, maintaining a success rate of over 92%. Under the condition of "resource failover", although the delay increases to 4.1 seconds, the system can complete redundant resource binding and substitution, maintaining overall stability. In "high concurrency scheduling", priority sorting and resource pooling mechanisms ensure a task success rate of over 90% and guarantee queue orderliness. In the context of "path constrained reconstruction", although the success rate decreased to 88.3%, the system still maintained stable water supply by generating suboptimal paths without interruption. Disturbance experiments were repeated 100 times under varying seeds and environment perturbations, with variance reported as mean ± SD.

6.3 System resource cost and optimization

The large-scale promotion of scheduling models driven by deep reinforcement learning in agricultural irrigation scenarios depends crucially on their adaptability in terms of computing resources, communication bandwidth, and hardware environment. Therefore, this study quantitatively evaluated the resource expenditure of the model under typical multi plot irrigation conditions and proposed optimization strategies. The model consists of three modules: edge perception, central decision-making, and interactive feedback. The edge perception module is deployed on sensor nodes or gateways, responsible for collecting and processing soil moisture, meteorological, and pump valve data. Under the conditions of 5Hz sampling frequency and parallel monitoring of 50 farmland plots, the CPU utilization rate of a single node remains stable within 30%, with a memory requirement of approximately 800MB. It can run on common ARM embedded devices, avoiding dependence on high-end hardware. The central decision-making module is based on GPU to generate irrigation paths and perform reinforcement learning inference. The experiment shows that under 100 concurrent irrigation tasks, the average scheduling cycle is 2.4 seconds, with the model computation cost accounting for 65% of the total delay. Real time operation can be supported on medium GPUs at the RTX A2000 level. If hardware is limited, lightweight network pruning and parameter quantization methods can be used to reduce computation by about 40%, while maintaining stable output in CPU environments. The interactive feedback module is based on WebSocket to achieve virtual real synchronization and data visualization. At 720p resolution, the bandwidth requirement is 3.1Mbps and the communication delay is less than 150ms, which can meet the real-time requirements of agricultural IoT environment. If in a network restricted area, layered transmission and edge caching strategies can be used to further compress bandwidth consumption by 30%. In terms of cost, the overall investment of the system mainly consists of sensors, communication modules, and mid-range GPU servers. When deployed in thousands of acres of farmland, the total cost is lower than the average level of most commercial agricultural intelligent irrigation platforms. Meanwhile, the modular structure allows farmers to gradually expand nodes based on their scale, providing good flexibility.

6.4 Application value of intelligent scheduling system in agriculture

In the process of precision and intelligent transformation in modern agriculture, irrigation scheduling systems not only need to cope with complex conditions of multiple plots and crops, but also need to achieve efficient utilization under limited water resources. The deep reinforcement learning driven intelligent irrigation scheduling system proposed in this article, combined with environmental perception and dynamic optimization mechanisms, has demonstrated outstanding value in agricultural applications. In terms of operational efficiency, the model is improved through path optimization and water source allocation strategies to reduce water source competition and irrigation conflicts between plots. The experimental results showed that the irrigation response delay was compressed to within 2 seconds, and the water resource utilization rate remained above 88%, significantly improving the matching between irrigation rhythm and crop water demand. The system has strong fault tolerance, can identify sudden rainfall and sensor anomalies, and quickly reconstruct strategies after faults occur to ensure water supply continuity. Simulation data shows that the scheduling interruption rate has decreased by over 40%, the irrigation completion rate has increased to 93%, conflict alarms have significantly decreased, and the burden of operation and maintenance has been effectively alleviated. At the management level, the system relies on the agricultural Internet of Things and visualization platform to present the real-time distribution of soil moisture, valve status, and water source surplus, allowing management personnel to intuitively grasp the operation status of farmland and make data-driven decisions. As a result, the traditional reliance on manual experience has gradually shifted towards scientific management based on data analysis, significantly improving the transparency and controllability of agricultural production. The system compatibility further enhances its potential for promotion. The scheduling platform can be connected to farmland monitoring, water conservancy scheduling, and meteorological forecasting systems through standard protocols, supporting remote deployment and modular tailoring. It can adapt to diverse application scenarios from small-scale farmland to large-scale agricultural areas, avoiding duplicate construction and information silos, and demonstrating strong application value.

7 Conclusion

The agricultural irrigation intelligent scheduling system based on deep reinforcement learning proposed in this study, combined with multi-source data perception and real-time feedback mechanism, significantly improves water resource utilization and crop yield. In the experiment, the system performed well in multi plot and multi crop scenarios, Water utilization rose by 12.7% ± 1.4%, and crop yield by 8.3% ± 0.9%, compared with the baseline ($p < 0.05$). Compared with traditional threshold control methods, the system has higher flexibility and accuracy, and can dynamically optimize irrigation strategies and make rapid adjustments in case of sudden climate and equipment failures. The system forms a closed-loop control through real-time perception and feedback, ensuring efficient allocation of water resources and maintaining stable operation in complex environments. The modular architecture of the system enables it to have strong scalability and adapt to agricultural production needs of different scales. In the future, this system is expected to be applied in large-scale agricultural production, promoting the intelligent development of agriculture. However, there are still some shortcomings in the research, mainly including: firstly, the adaptability verification of the system under extreme climate conditions is limited; Secondly, in terms of data collection and system integration, there is still a need to address issues of data loss and hardware compatibility; Thirdly, in high concurrency scheduling scenarios, the response time of the system may be affected to some extent.The source code, trained policies, and a sanitized subset of the dataset are available from the corresponding author upon reasonable request.

References

[1] Saikai Y, Peake A, Chenu K. Deep reinforcement learning for irrigation scheduling using high-dimensional sensor feedback[J]. PLOS Water, 2023, 2(9): e0000169.https://doi.org/10.1371/journal.pwat.0000169

[2] Alibabaei K, Gaspar PD, Assunção E, Alirezazadeh S, Lima TM. Irrigation optimization with a deep reinforcement learning model: Case study on a site in Portugal[J]. Agricultural Water Management,2022,263:107480.https://doi.org/10.1016/j.agwat.2022.107480

[3] Yang X, Hu J, Porter D, Marek T, Heflin K, Kong H. Deep reinforcement learning-based irrigation scheduling[J]. Transactions of the ASABE, 2020, 63(3): 549–556.https://doi.org/10.13031/trans.13633

[4] Ding X , Du W .Optimizing Irrigation Efficiency using Deep Reinforcement Learning in the Field[J].ACM Transactions on Sensor Networks, 2024, 20(4).https://doi.org/10.1145/3662182

[5] Chen M, Cui Y, Wang X, et al. A reinforcement learning approach to irrigation decision-making for rice using weather forecasts[J]. Agricultural Water Management,2021,250:106838.https://doi.org/10.1016/j.agwat.2021.106838

[6] Jimenez AF, Cardenas PF, Jimenez F, et al. A cyber-physical intelligent agent for irrigation scheduling in horticultural crops[J]. Computers and Electronics in Agriculture,2020,178:105777.https://doi.org/10.1016/j.compag.2020.105777

[7] Alves RG, Maia RF, Lima F. Development of a digital twin for smart farming: Irrigation management system for water saving[J]. Journal of Cleaner Production, 2023,388:135920.https://doi.org/10.1016/j.jclepro.2023.135920

[8] Zia H, Rehman A, Harris NR, et al. An experimental

comparison of IoT-based and traditional irrigation scheduling on a flood-irrigated subtropical lemon farm[J]. Sensors,2021,21(12):4175.https://doi.org/10.3390/s21124175

[9] Kelly TD, Foster T, Schultz DM. Assessing the value of deep reinforcement learning for irrigation scheduling[J]. Smart Agricultural Technology, 2024, 1: 100403. https://doi.org/10.1016/j.atech.2024.100403

[10] Liu K,Jiao X,Guo W,Gu Z,Li J. Improving irrigation performance by using adaptive border irrigation with deep reinforcement learning[J].Agronomy,2023,13(12):2907.https://doi.org/10.3390/agronomy13122907

[11] Mai Z, Zhang L, Li X, et al. multi-objective modeling and optimization of water-saving irrigation scheduling using deep reinforcement learning[J]. Agricultural Water Management,2024,108959.https://doi.org/10.1016/j.agwat.2024.108959

[12] Chen Y, Lin M, Yu Z, Sun W, Fu W, He L. Enhancing cotton irrigation with distributional actor–critic reinforcement learning[J]. Agricultural Water Management,2024,307:109194.https://doi.org/10.1016/j.agwat.2024.109194

[13] Agyeman B T, Nouri M, Appels W M, Liu J, Shah S L. Learning-based multi-agent MPC for irrigation scheduling[J]. Control Engineering Practice, 2024, 147: 105908. https://doi.org/10.1016/j.conengprac.2024.105908

[14] Agyeman B T, Decard-Nelson B, Liu J, Shah S L. A semi-centralized multi-agent RL framework for efficient irrigation scheduling[J]. arXiv preprint, 2024, preprint.https://doi.org/10.48550/arXiv.2408.08442

[15] Madondo M, Azmat M, Dipietro K, et al. A SWAT-based reinforcement learning framework for crop management[J]. arXiv preprint,2023, preprint.https://doi.org/10.48550/arXiv.2302.04988

[16] Liu J., Yang J., Jie X., Chang F., Ma L., Su H. Optimizing irrigation scheduling using Deep Reinforcement Learning and crop growth model [C]. 2025 ASABE Annual International Meeting, Paper No. 2500569. https://doi.org/10.13031/aim.202500569

[17] Kåge L, Milić V, Andersson M, Wallén M. Reinforcement learning applications in water resource management: a systematic literature review[J]. Frontiers in Water, 2025,7:1537868.https://doi.org/10.3389/frwa.2025.1537868

[18] Jia B. Reservoir Irrigation Operation Design Based on Dijkstra Algorithm Combined with ACO Algorithm[J]. Informatica,2024,48(12): 171–184.https://doi.org/10.31449/inf.v48i12.6005

[19] Xiao L. FD3QN: A Federated Deep Reinforcement Learning Approach for Cross-Domain Resource Cooperative Scheduling in Hybrid Cloud Architecture[J]. Informatica, 2025, 49(10): 127–146.https://doi.org/10.31449/inf.v49i10.7114

[20] Hajgató G, Paál G, Gyires-Tóth B. Deep Reinforcement Learning for Real-Time Optimization of Pumps in Water Distribution Systems. Journal of Water Resources Planning and Management, 2020,146(11): 04020079.https://doi.org/10.1061/(ASCE)WR.1943-5452.0001287