Journal Pre-proof

Knowledge guided fuzzy deep reinforcement learning

Peng Qin, Tao Zhao

PII: S0957-4174(24)02690-3
DOI: https://doi.org/10.1016/j.eswa.2024.125823
Reference: ESWA 125823

To appear in: *Expert Systems With Applications*

Received date : 4 August 2024
Revised date : 22 October 2024
Accepted date : 15 November 2024

Please cite this article as: P. Qin and T. Zhao, Knowledge guided fuzzy deep reinforcement learning. *Expert Systems With Applications* (2024), doi: https://doi.org/10.1016/j.eswa.2024.125823.

Revised manuscript (with changes marked)

Knowledge Guided Fuzzy Deep Reinforcement Learning

Peng Qin[a], Tao Zhao[a,*]

[a]College of Electrical Engineering, Sichuan University, Chengdu 610065, China

ARTICLE INFO

Keywords:
Knowledge guide
Fuzzy system
Reinforcement learning
Deep Q-network

ABSTRACT

Reinforcement learning (RL) addresses complex sequential decision-making problems through interactive trial-and-error and the handling of delayed rewards. However, reinforcement learning typically starts from scratch, necessitating extensive exploration, which results in low learning efficiency. In contrast, humans often leverage prior knowledge to learn. Inspired by this, this paper proposes a semantic knowledge-guided reinforcement learning method (KFDQN), which fully utilizes knowledge to influence reinforcement learning, thereby improving learning efficiency, training stability, and performance. In terms of knowledge representation, considering the strong fuzziness of semantic knowledge, a fuzzy system is constructed to represent this knowledge. In terms of knowledge integration, a knowledge-guided framework that integrates a hybrid action selection strategy (HYAS), a hybrid learning method (HYL), and knowledge updating is constructed in conjunction with the existing reinforcement learning framework. The HYAS integrates knowledge into action selection, reducing the randomness of traditional exploration methods. The HYL incorporates knowledge into the learning target, thereby reducing uncertainty in the learning objective. Knowledge updating ensures that new data is utilized to update knowledge, avoiding the negative impact of knowledge limitations on the learning process. The algorithm is validated through numerical tasks in OpenAI Gym and real-world mobile robot Goal Reach and obstacle avoidance tasks. The results confirm that the algorithm effectively combines knowledge and reinforcement learning, resulting in a 28.6% improvement in learning efficiency, a 19.56% enhancement in performance, and increased training stability.

1. Introduction

Since its inception, RL has achieved tremendous success in areas such as Go and gaming, demonstrating its powerful learning capabilities and vast potential for application (Li, 2023). With the continuous development of reinforcement learning, it has been widely applied in various fields such as intelligent driving (Kiran, Sobh, Talpaert, Mannion, Sallab, Yogamani and Pérez, 2022; Wang, Chen, Sun and Sun, 2024), robot control (Kaufmann, Bauersfeld, Loquercio, Müller, Koltun and Scaramuzza, 2023; Wei, Zhou, Zhu, Ma and Ma, 2023), and healthcare (Chen, Qiu, Tan, Fang and Jin, 2022; Yala, Mikhael, Lehman, Lin, Strand, Wan, Hughes, Satuluru, Kim, Banerjee et al., 2022). However, due to the interactive nature of reinforcement learning with its environment, agents require large amounts of data and numerous trial-and-error processes for learning, which leads to low sample efficiency. Furthermore, in environments with sparse or delayed rewards, agents face difficulties in obtaining effective rewards, leading to challenges in model convergence. Knowledge-guided reinforcement learning has emerged as a promising approach to mitigate these limitations.

The low learning efficiency in traditional RL methods is primarily due to trial and error, reflected in the following: 1) The random exploration mechanism involves a large number of uncertain action selections, making it difficult to quickly find the correct action; 2) The initial parameters of intelligent agents are usually random and require extensive data for training, but effective data is difficult to obtain in the early exploration stages. In addition, an important factor affecting training stability is that the construction of target values depends on non-optimal models. These challenges directly or indirectly impact the model's overall performance.

To address these issues, the motivation of this study is to take inspiration from human learning methods and leverage the advantages of prior knowledge to construct a semantic knowledge-guided reinforcement learning method that improves learning efficiency and stability, thereby enhancing model performance. This study primarily addresses the following issues: 1) knowledge representation and integration with reinforcement learning; 2) action selection to reduce uncertainty in decision-making; and 3) construction of reasonable target values to minimize uncertainty in the learning process. The main contributions are as follows:

- 1) This paper draws on human learning patterns (starting with supervised learning, followed by autonomous learning), expands the traditional DQN framework, and integrates fuzzy systems, knowledge updates, and guidance into the existing structure. This supports the representation of knowledge and its integration with reinforcement learning.
- 2) To improve the low learning efficiency inherent in the trial-and-error mechanism, we propose a hybrid action selection strategy that reduces the randomness in action selection, thereby enhancing learning efficiency. Additionally, to mitigate the instability encountered during the learning process, we introduce a hybrid learning method that integrates the influence of

*Corresponding author.
qinpeng2@stu.scu.edu.cn (P. Qin); zhaotaozhaogang@scu.edu.cn (T. Zhao)
ORCID(s):

the knowledge system, improving the overall stability of the learning process.

- 3) To verify the effectiveness of the proposed method, we conducted experiments in both standard environments such as CartPole and MountainCar, as well as in real-world settings involving mobile robots performing Goal Reach and obstacle-avoidance tasks. The experimental results demonstrate significant improvements in both learning efficiency and performance compared to baseline reinforcement learning algorithms, validating the practicality and feasibility of the proposed method.

1.1. Threat to validity

Despite the promising results demonstrated by the proposed KFDQN method in enhancing learning efficiency, stability, and performance, several potential threats to its validity should be acknowledged: 1. The rules of the fuzzy system must align with the knowledge of the task, which can pose challenges in capturing all dimensions of knowledge when dealing with high-dimensional complex tasks; therefore, identifying and selecting key feature dimensions for knowledge construction is essential. 2. While the effectiveness of the method has been validated in various tasks, including CartPole, MountainCar, and real-world mobile robot Goal Reach and obstacle avoidance, further research is needed to assess its generalizability in more complex environments. 3. Although the focus of this study was not primarily on interpretability, exploring the internal transparency of the system could enhance understanding of its decision-making processes and strengthen its practical applicability.

2. Related work

2.1. Reinforcement learning

RL has been widely applied to decision-making problems. Traditional methods such as Q-learning and SARSA have demonstrated effectiveness in simple tasks; however, they rely on discrete state spaces and are prone to the curse of dimensionality, making them unsuitable for environments with continuous state spaces. The introduction of deep learning into RL, particularly through the development of Deep Q-Networks by (Mnih, Kavukcuoglu, Silver, Rusu, Veness, Bellemare, Graves, Riedmiller, Fidjeland, Ostrovski et al., 2015), significantly improved the scalability of RL methods by allowing the use of deep neural networks as function approximators. While DQN achieved remarkable success in playing Atari games, it suffers from low learning efficiency and instability during training, especially in environments with sparse or delayed rewards. Therefore, improving learning efficiency and stability is an important demand and challenge for RL. To this end, scholars have conducted extensive research, including introducing curiosity exploration mechanisms with similarity curiosity modules (Chen, Zhai, Gao, Xu, Yang, Li, Ding, Feng and Wang, 2023; Sun, Fang, Zheng and Tang, 2022) and probabilistically regenerating the exploration space to constrain the sample optimization methods of exploration behavior (Xu, Zhu, Liu and Zhao, 2021). They also use integrated proximal policy optimization, adopt diverse regularization to promote the exploration of policy integration methods (Yang, Ren, Luo, Liu, Liu, Bian, Zhang and Li, 2022), construct environment models, reduce real-world samples through model-accelerated learning methods(Guan, Ren, Li, Sun, Luo and Li, 2020; Wu, Huang and Lv, 2022), and utilize past experience for meta-learning and transfer learning (Zhang, Wu, Zhang and Wang, 2022; Zhu, Lin, Jain and Zhou, 2023).

2.2. Knowledge-Guided reinforcement learning

To improve the efficiency and stability of RL, many studies have proposed knowledge-guided reinforcement learning (KGRL), which aims to enhance the learning process by integrating external knowledge or domain-specific information. These methods include behavioral cloning (Lin, He, Li, Liu, Wang and Zhu, 2024; Wang, Fernandez and Stiller, 2022), where, for example, (Wang et al., 2022) records real traffic trajectories and learns driving strategies from the recorded data using behavioral cloning. Additionally, inverse reinforcement learning (IRL) has been used to infer reward functions from behavior to guide agent training (Pitombeira-Neto, Santos, da Silva and de Macedo, 2024; Song, Li and Xu, 2022; Zhao, Wang and Wang, 2023). For instance, (Zhao et al., 2023) proposes a moth-flame optimization algorithm for IRL, which learns a reward function from expert demonstrations and applies it to Q-learning. Similarly, (Pitombeira-Neto et al., 2024) models the reward function as a parametric deterministic function of observable features through IRL and uses it for agent learning. These methods are based on demonstration data, which depend on expert demonstrations typically collected from specific environments, representing a low-level form of knowledge representation. Moreover, knowledge has been incorporated into various pipelines such as model construction and value function modeling. For instance,(Chiou, 2024) employed knowledge to build models,(Du, Chen, Zhao, Liao and Zhu, 2023) integrated knowledge into the learning environment, while (Gao, Si and Huang, 2023) utilized knowledge for value function construction. Additionally, distributed reinforcement learning is a type of guided RL, typically drawing on expert knowledge and leveraging Parallel Learning within the Learning Strategy pipeline(Eßer, Bach, Jestel, Urbann and Kerner, 2023). (Espeholt, Soyer, Munos, Simonyan, Mnih, Ward, Doron, Firoiu, Harley, Dunning et al., 2018) proposed an importance-weighted distributed agent learning framework to facilitate positive transfer across different tasks. Although these methodologies present diverse perspectives on knowledge representation and embedding, achieving significant success, the involved knowledge is generally domain-specific and expert-level, often represented in a data-driven or implicitly embedded manner. This poses challenges for interpretation and modification.

2.3. Fuzzy knowledge representation

In contrast, semantic knowledge refers to the meaning of language, including concepts, relationships, and other

related aspects. It emphasizes the understanding of natural language, is inherently abstract, and provides greater adaptability and flexibility. The aforementioned methods are not well-suited for representing and utilizing such knowledge. Given the inherent vagueness and uncertainty of semantic knowledge, traditional methods such as classical binary logic rules cannot adequately support approximate reasoning models. In contrast, fuzzy logic and fuzzy sets, proposed by (Zadeh, 1965, 1992), can transform vague semantic concepts into precise mathematical representations. Using a rule-based format, they are capable of expressing various types of semantic knowledge. The reliability of fuzzy systems has been validated in fields such as prediction (Mei, Zhao and Xie, 2024), classification (Gu, 2023), and control (Zhao, Qin, Dian and Guo, 2024), demonstrating the feasibility of using fuzzy systems to represent semantic knowledge.

3. Preliminary knowledge

3.1. Fuzzy system

"Human natural language often embodies ambiguity when describing the attributes of an object, such as characterizing a person's height as 'tall' or 'short,' or their physique as 'overweight' or 'slim.' These ambiguous descriptions pervade every aspect of human daily life, constituting crucial experiential knowledge for managing everyday activities. For instance, such experiential knowledge exists when driving a car: 'If the road ahead is exceedingly narrow, decelerate.' This experiential knowledge holds significant value, and a fuzzy system can aptly articulate it while executing precise mathematical computations. The representation of semantic knowledge in fuzzy systems primarily relies on fuzzy sets and fuzzy rules. Fuzzy sets come in various forms, including Gaussian and triangular, but all represent the degree of attributes through membership functions. Among these, the Gaussian form is widely utilized, with its mathematical expression and graphical representation shown in equation (1) and Fig. 1, respectively (Mendel, 2017)."

$$\mu_A(x) = exp[-\frac{1}{2}\left(\frac{x-c}{\sigma_1}\right)^2] \tag{1}$$

Where A denotes the name of the fuzzy set, x represents the input of the fuzzy set, σ_1 signifies the width of the fuzzy set, and c corresponds to the center of the fuzzy set.

Fuzzy rules express the relationships between attributes described by fuzzy sets in an IF-THEN format. For a fuzzy system, there will be several rules, taking the kth rule as an example, its expression is (2)

$$\begin{aligned} &\mathbb{R}^k\text{: if } x_1 \text{ is } A_1^k \text{ and ... and } x_n \text{ is } A_n^k \\ &\text{then } y_1 \text{ is } B_1^k \text{ and } y_2 \text{ is } B_2^k \text{ ... } y_n \text{ is } B_n^k \quad k = 1, ..., N \end{aligned} \tag{2}$$

where $x_i, (i = 1, 2, \dots, n)$ represents the input, ${A_i}^k$ denotes the fuzzy set corresponding to the ith input, y_i represents the output, and ${B_i}^k$ signifies the fuzzy set corresponding to the ith output.

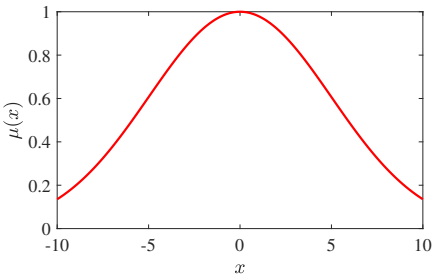


Figure 1. The gaussian fuzzy set

Furthermore, the fuzzy system encompasses an inference engine and defuzzification. The inference engine can be realized by various operators, typically employing the product operator to compute the activation strength of the rule. For instance, considering the kth rule, which is defined as (3) .

$$f^k = \mu_{{A_1}^k}(x_1) * \mu_{{A_2}^k}(x_2) * \cdots * \mu_{{A_3}^k}(u_n) \tag{3}$$

Defuzzification is the process of transforming fuzzy values into precise and clear values to obtain the output of the fuzzy system. For center average defuzzification, it is represented as (4)

$$y = \frac{\sum_{k=1}^{N} \theta_k f^k}{\sum_{k=1}^{N} f^k} \tag{4}$$

where y is output of fuzzy system, θ_k is the center of the consequent fuzzy set, and Nrepresents the total number of rules.

3.2. Reinforcement learning basics

RL primarily addresses the sequential decision-making problems inherent in dynamic systems, necessitating a series of actions to attain a goal. Such problems are typically defined as Markov Decision Processes (MDP), characterized by a quintuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$, where $\mathcal{S}$ denotes the set of system states, $\mathcal{A}$ signifies the set of actions, $\mathcal{P}(s'|s, a_t)$ is the state transition function, defining the probability of transitioning from state s to s' under action a_t, r is the artificially designated reward function, indicating the immediate reward acquired from an action, and $\gamma \in (0, 1]$ is the discount factor for subsequent rewards. To reach the ultimate objective, it is necessary to select a series of actions in various states to facilitate state transitions, and the method of selecting actions in each state is termed a policy, represented by $\pi(\text{a}|\text{s})$. The optimal policy is one that maximizes the expected reward, thus the objective of RL is (5).

$$\pi^* = \underset{\pi}{\text{argmax}} \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)] \tag{5}$$

One method to solve for π^* involves the iterative process over the state value function and the action value function, subsequently deducing the optimal policy. The Bellman expressions for the state value function and the action value function are denoted as (6) and (7), respectively(Li, Wang, Zheng, Jin, Huang, Zhang and Wang, 2022).

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}_{\pi}[r(s,a) + \gamma V^{\pi}(s')] \\ &= \sum_{a\in A} \pi(a|s) \left(r(s,a) + \gamma \sum_{s'\in S} p(s'|s,a) V^{\pi}(s') \right) \end{aligned} \tag{6}$$

$$\begin{aligned} Q^{\pi}(s,a) &= \mathbb{E}_{\pi}[r(s,a) + \gamma Q^{\pi}(s',a')] \\ &= r(s,a) + \gamma \sum_{s'\in S} p(s'|s,a) \sum_{a'\in A} \pi(a'|s') Q^{\pi}(s',a') \end{aligned} \tag{7}$$

The aforementioned state value function and action value function incorporate the state transition function, implying that the MDP process is known. Under this condition, the agent does not actually need to interact with the environment to sample data. However, in most scenarios, the state transition function is unknown, which requires interaction with the environment and learning from the data obtained through sampling. This type of learning method is referred to as model-free learning. The method proposed in this paper is based on the fuzzy system and DQN, and also falls within the category of model-free learning.

3.3. Deep Q-network

Deep Q-network(DQN) and traditional Q-Learning share fundamental similarities. In conventional Q-Learning, a table (Q-table) is employed to store the Q-values for each state-action pair. However, in complex environments where the state space can be exceedingly large or even continuous, table-based methods become impractical. Consequently, function approximation is commonly adopted, leveraging the powerful fitting capability of neural networks to represent the state-value function. In this context, DQN utilizes a deep neural network, denoted as $Q_{\omega}(s,a)$ (ω is the parameters of network), to serve as the action-value function, thereby equipping it with the capacity to handle problems with high-dimensional state spaces(Mnih et al., 2015). In Q-Learning, temporal difference methods are employed to update the action-value function, with the update rule expressed as (8) where α represents the step size for updating the value estimate, which can be taken as a constant.

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r(s,a) + \gamma \max_{a'\in\mathcal{A}} Q(s',a') - Q(s,a) \right] \tag{8}$$

As DQN utilizes a neural network to approximate the action-value function, its learning objective is to identify the optimal parameters, denoted as ω^*, such that $Q_{\omega}(s,a)$ approximates the target value $r(s,a) + \gamma \max_{a'} Q\omega(s'_i, a')$. The optimization function is denoted as (9).

$$\omega^* = \arg\min_{\omega} \frac{1}{2N} \sum_{i=1}^{N} \left[Q_{\omega}(s_i,a_i) - \left(r_i + \gamma \max_{a'} Q_{\omega}(s'_i,a') \right) \right]^2 \tag{9}$$

To satisfy the independence assumption of samples and enhance sample efficiency, DQN incorporates an "experience replay" mechanism. This specific implementation entails maintaining an experience pool, storing the four-tuple data (s_i, a_i, r_i, s'_i) sampled from the environment at each instance into the replay buffer, and subsequently randomly sampling several data points from the replay buffer for training the $Q_{\omega}(s,a)$. Furthermore, the ultimate update objective of the DQN algorithm is to approximate $Q_{\omega}(s,a)$ to $r(s,a) + \gamma \max_{a'} Q_{\omega}(s'_i,a')$. Given that the network Q_{ω} itself is included in the error during training, fluctuations may occur, thus the network in the optimization function is replaced with another target network Q_{ω^-}, as depicted in the equation.

4. Knowledge guided fuzzy deep Q-network

4.1. Overall structure

The overall framework of KFDQN, as shown in the Fig.2, primarily comprises a value network, fuzzy system, HYAS, and HYL. This framework draws inspiration from human learning patterns (students passively receive knowledge imparted by mentors, learn from predecessors, and form a preliminary foundation. Then, based on this foundation, they engage in autonomous extended learning with the assistance of mentors), and is divided into two learning stages. In the first stage, the fuzzy system (mentor) interacts directly with the environment, and the value network (student) learns supervisedly from the data generated by the fuzzy system's interaction, thereby acquiring knowledge. In the second stage, the value network and the fuzzy system interact with the environment together through HYAS, and learn under the guidance of HYL.Ultimately, a cyclical learning framework is formed, where the fuzzy system guides the agent to interact and learn with the environment, the fuzzy system learns itself, and the fuzzy system provides further guidance. The specific features of this method are listed as follows.

Remark 1. *Feature 1: Incorporation of Semantic Knowledge.*

Behavior cloning and inverse learning represent knowledge through expert behavior data and directly use it for learning. However, the acquisition and preparation of expert behavior data are not easy. In contrast, semantic knowledge is widely distributed and easily obtained directly from human experience. Therefore, this paper uses a fuzzy system to represent semantic knowledge and integrates it into the learning of the agent.

Compared with using behavioral data, this method has the advantage of easy knowledge acquisition due to the fact

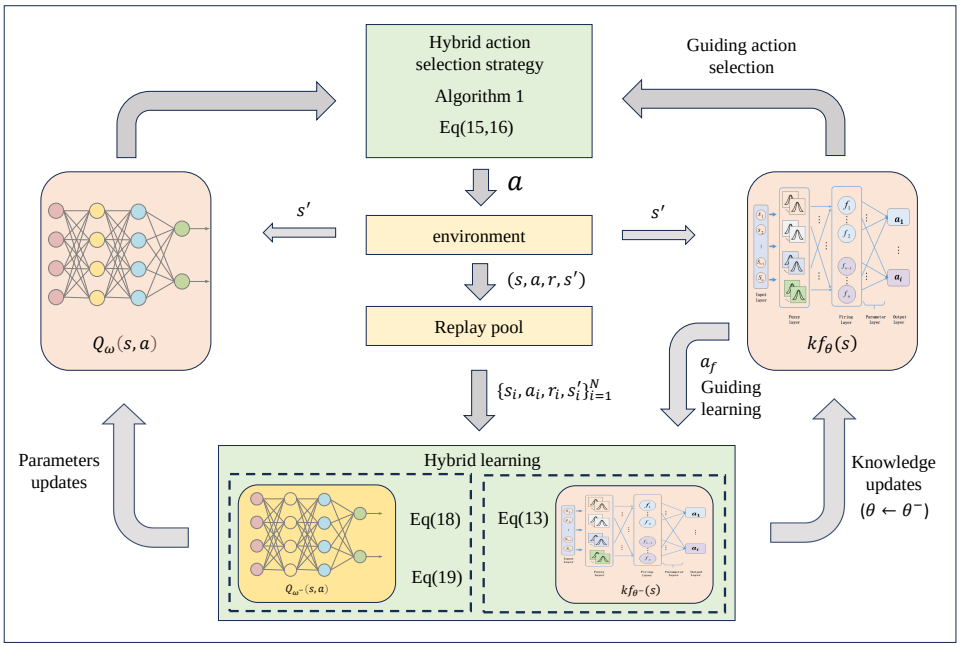


Figure 2. The diagram of general structure of KFDQN

that semantic knowledge derived from human experience is easier to obtain. In addition, due to the wide adaptability of semantic knowledge (the same semantic knowledge can be applied to different tasks), this method can reuse the same knowledge to adapt to diverse tasks.

Remark 2. *Feature 2: Knowledge-Guided Action Selection.*

The traditional $\varepsilon - greedy$ action selection strategy selects actions randomly with a certain probability, which may lead to a large number of invalid states in the training process, resulting not only in low data collection efficiency but also poor data quality. In contrast, the HYAS proposed in this paper includes two stages. The first stage involves direct interaction between the fuzzy system and the environment, quickly obtaining effective data, and providing support for the supervised learning of the value network. In the second stage, the fuzzy system and the value network interact with the environment together, reducing the uncertainty of deriving actions from a single model. This alleviates the problems of low data collection efficiency, poor data quality, and randomness of action selection, thereby improving training efficiency.

Remark 3. *Feature 3: Knowledge-Guided Learning.*

The traditional temporal difference learning target uses a greedy method and a value network to calculate the value of subsequent states. Since the value network is not optimal, the learning target obtained in this way has a high degree of uncertainty. In contrast, the HYL proposed in this paper uses the action output of the knowledge system to balance the value estimation of the next state, reducing the uncertainty of the greedy method and the original value network in calculating the value of subsequent states, thereby reducing the uncertainty of the learning target and enhancing the stability of the training process.

4.2. Construction of semantic knowledge fuzzy system

4.2.1. Construction of rules

The construction of a semantic knowledge fuzzy system is the representation of the prior knowledge that humans possess for completing a specific task, in the form of a fuzzy system. The primary step involves the transformation of this prior knowledge into fuzzy rules. Taking the inverted pendulum as an example, human prior knowledge can be semantically described as: 'When the pendulum leans to the left, the cart needs to move to the left; when the pendulum leans to the right, the cart needs to move to the right.' Here, 'left' and 'right' are fuzzy semantic concepts, necessitating the establishment of corresponding fuzzy sets 'left' and 'right.' The semantic logic can be directly transformed into

an IF-THEN format as (10).

$$\begin{cases} if\ p\ is\ left\ then\ v\ left \\ if\ p\ is\ right\ then\ v\ right \end{cases} \tag{10}$$

However, for a broader range of systems, the dimensions of input and output may be high-dimensional, thus the construction of a more extensive semantic fuzzy system needs to be based on specific requirements. As the semantic knowledge fuzzy system needs to be integrated with DQN, another key to constructing the semantic fuzzy system is to transform the prior knowledge into the relationship between states and actions. The final form of the rule is as (11).

$$\begin{cases} IF\ s_1\ is\ IF_1^1\ AND\ s_2\ is\ IF_2^1\ ...s_n\ is\ IF_n^1\ THEN\ a_1\ is\ OF_1^1\ AND\ a_2\ is\ OF_2^1\ ...a_n\ is\ OF_n^1 \\ IF\ s_1\ is\ IF_1^2\ AND\ s_2\ is\ IF_2^2\ ...s_n\ is\ IF_n^2\ THEN\ a_1\ is\ OF_1^2\ AND\ a_2\ is\ OF_2^2\ ...a_n\ is\ OF_n^2 \\ ... \\ IF\ s_1\ is\ IF_1^N\ AND\ s_2\ is\ IF_2^N\ ...s_n\ is\ IF_n^N\ THEN\ a_1\ is\ OF_1^N\ AND\ a_2\ is\ OF_2^N\ ...a_n\ is\ OF_n^N \end{cases} \tag{11}$$

Where, $S = [s_1, s_2 \dots s_n]$ represents the system states, IF_i^K is the fuzzy set corresponding to state s_i in the kth rule , and similarly, $OF_i^k$is the fuzzy set for each action a_i in the kth rule. Notably, the output of each rule covers the entire action space and outputs the score for each action. The final output action is selected by comparing these scores. This approach ingeniously resolves the contradiction between the continuous output of the fuzzy system and the discrete output of the DQN, and allows the fuzzy system to perform online parameter learning through state data pairs. Based on the rule base of (2) and using the product operator of (3), the final semantic fuzzy system can be obtained as (12).

$$kf_\theta(s) = \theta\zeta(s) \tag{12}$$

4.2.2. Knowledge update

The semantic knowledge fuzzy system is built upon human experiential knowledge. Given the inherent limitations in human cognition, experiential knowledge itself is characterized by uncertainty. This implies that different individuals may construct different fuzzy systems for the same task, reflected in the differences in fuzzy rules and parameters. To obtain a more comprehensive and accurate semantic knowledge fuzzy system, and to provide effective guidance during the learning process of the Q network, it is necessary to update the parameters of the semantic knowledge fuzzy system, thereby achieving knowledge updating. For this purpose, this paper designs a knowledge updating method based on the characteristics of the data.

The method of knowledge updating primarily involves the collection of state-action data from the experience pool to update the consequent parameters of the fuzzy system. To prevent the instability in training caused by frequent parameter updates, we borrow the concept of target networks from DQN, constructing two fuzzy systems, $kf_\theta(s)$ and $kf_{\theta^-}(s)$. Here, $kf_\theta(s)$ is used to guide the learning of the Q network, while $kf_{\theta^-}(s)$ is used to learn new knowledge from the data. After several episodes, the parameters of $kf_\theta(s)$ are updated with those of $kf_{\theta^-}(s)$. Since (s, a) can be regarded as input and label respectively, the update of fuzzy system parameters adopts a supervised learning approach. Furthermore, as the output of the fuzzy system is the score of each action and a is a specific action, we employ cross-entropy to construct the loss function, as (13).

$$\mathcal{L}(\theta^-) = \mathbb{E}_{(\mathbf{a},\mathbf{s})\sim D^E}[\log(softmax[kf_{\theta^-}(s)]) * a] \tag{13}$$

4.3. Utilize knowledge

Similar to other RL methods, the core of the KFDQN algorithm still includes two parts: exploration and exploitation. The methods of exploration and exploitation significantly influence the learning process and the final policy. Therefore, how to integrate prior knowledge into both exploration and learning is key to constructing KFDQN.

4.3.1. Hybrid action selection strategy

The DQN algorithm employs the $\varepsilon - greedy$ method for exploration, which implies that there is a certain probability ε for random action selection, and with a probability of $1 - \varepsilon$, actions are selected based on the maximum value of the action-value function. This approach enables random exploration with a certain probability, as expressed in (14).

$$\pi(a|s) = \begin{cases} 1 - \varepsilon & a = argmaxQ(s, a) \\ \varepsilon & a = rand_a \end{cases} \tag{14}$$

While this method is simple to implement, pure random exploration may lead to significant fluctuations in the learning process. Moreover, there is uncertainty in selecting actions based on the maximum value of the action-value function before the model reaches its optimum. This is because the selected action cannot be guaranteed to be the optimal action for the current state before the model is optimized. Therefore, to reduce the impact of random selection and model uncertainty on exploration, this paper integrates the semantic knowledge fuzzy system with the output of the value network. The aim is to avoid random exploration in the early guidance stage and weaken model uncertainty in the later reinforcement learning stage, thereby proposing a hybrid action selection strategy, as shown in Algorithm 1.

Where, p is the random probability value, $s = [s_1, s_2 \dots s_n]$ represents the system state, $episodes\_cur$ is the current

Algorithm 1 Hybrid action selection strategy

Require: p, s,ϵ,ep_t
Output: $action$
if $p < \epsilon$ **then**
  $action = a_f$
else
  if $episodes\_cur < ep_t$ **then**
    $action = a_f$
  else
    $action = hy_a$
  end if
end if
return $action$

number of episodes, and ep_t is the episode threshold. This parameter is adjustable and is used to divide the learning stage. a_f is the action output of the semantic knowledge fuzzy system, and its expression is as (15):

$$a_f = \underset{i\in[0,1...n]}{\text{argmax}}(softmax[kf_\theta(s)]) \tag{15}$$

hy_a is a hybrid action, and its expression is (16).

$$hy_a = \underset{i\in[0,1...n]}{argmax}(h_1 * softmax[kf_\theta(s)]+h_2 * softmax[Q(s)]) \tag{16}$$

Where, h_1 and h_2 are adjustable parameters used to regulate the intervention strength of the semantic fuzzy system on the action selection of the Q network in the later stage.

4.3.2. Hybrid learning method

The learning of the $Q_\omega(s_i, a_i)$ network in DQN is similar to that in Q-Learning, both of which use the Bellman equation to update network parameters, as shown in (9). The term $\underset{a'}{max}Q_\omega(s'_i, a')$embodies the concept of greediness, selecting the action a' that maximizes Q_ω at state s'. However, a' is only the current optimal action under state s', and may not necessarily be the optimal action under state s', which could lead to fluctuations during model training. If we use different types of external models to provide the reference optimal action under state s', and construct the learning target according to a certain weight, this uncertainty can be reduced. Based on this idea, this paper proposes a HYL, incorporating the highest value action a_f of the semantic fuzzy system under s' as part of the learning target construction. The final learning target function is (17).

$$\omega^* = \arg\min_{\omega}(\frac{1}{2}[Q_\omega(s,a) - (r + m * \gamma \underset{a'}{max}Q_{\omega^-}(s', a') + n * \gamma Q_\omega(s', a_f))]^2) \tag{17}$$

Where, m and n are adjustable parameters with the condition that $m + n = 1$. The objective function for batch data training through the experience pool is (18).

$$\omega^* = \arg min\frac{1}{2N}\sum_{i=1}^{N}[Q_\omega(s_i,a_i) - (r_i + m * \gamma \underset{a'}{max}Q_{\omega^-}(s'_i, a') + n * \gamma Q_\omega(s'_i, a_{f_i}))]^2 \tag{18}$$

During the initial phase, the role of the fuzzy system is dominant, and the data generated in this phase holds significant reference value. Therefore, in this phase,$Q_\omega(s)$ adopts a supervised learning approach to rapidly learn the characteristics of the fuzzy system from the data it generates. Its learning method is similar to that of the fuzzy system, also employing cross-entropy to construct the loss function, using the output of the fuzzy system as the label, as shown in (19).

$$\mathcal{L}(\theta^-) = \mathbb{E}_{(\mathbf{s})\sim D^s}[\log(softmax[Q_\omega(s)]) * \underset{a}{\text{argmax}}(kf_\theta(s))]] \tag{19}$$

4.4. KFDQN algorithm

In summary, the algorithmic process of KFDQN can be divided into two parts: supervised learning and reinforcement learning, each of which is built on the basis of interaction with the environment. Different action selection strategies and parameter update methods are adopted at different stages. The final process is as shown in Algorithm 2.

Algorithm 2 KFDQN

Construct fuzzy rules based on knowledge and set the parameter θ of fuzzy system $f_\theta(s)$.
Initialize network $f_{\theta^-}(s)$ with random parameter θ^-.
Initialize network $Q_\omega(s, a)$ with random parameter ω.
Copy the same parameter $\omega^- \leftarrow \omega$ to initialize $Q_{\omega^-}(s, a)$.
Initialize experience replay pool R
for episode = 1, 2, ..., M **do**
  Obtain the initial state of the environment s_0
  for t = 1, 2, ..., T **do**
    Based on the current network $Q_\omega(s, a)$, select action a_t using $HYAS$.
    Execute action a_t to receive reward r_t and the next state s_{t+1}.
    Store (s, a, r, s_{t+1}) in the experience replay pool R
    if The data in R is sufficient **then**
      Randomly sample N pieces of data from R, $\{s_i, a_i, r_i, s_{i+1}\}_{i=1}^{N}$.
      if $episodes\_cur < ep_t$ **then**
        Update $Q_\omega(s, a)$ base on (19).
      else
        Update $Q_\omega(s, a)$ base on (18).
      end if
      update $kf_{\theta^-}(s)$ base on (13)
      if $episodes_cur/C == 0$ **then**
        Update $Q_{\omega^-}(s, a)$ based on $\omega^- \leftarrow \omega$
        Update kf_θ based on $\theta \leftarrow \theta^-$
      end if
    end if
  end for
end for

4.5. Computational efficiency analysis

In different algorithms, the total number of parameters primarily depends on the input state dimension, action dimension, and network architecture. Suppose the input state dimension is s_n and the action dimension is a_n. DQN uses a deep neural network to approximate the Q-function. If the network consists of an input layer, L hidden layers, and an output layer, where each hidden layer has n_i neurons, the total number of parameters for a single Q-network is:

$$P_{DQN} = s_n \cdot n_1 + n_1 + \sum_{i=2}^{L} (n_{i-1} \cdot n_i + n_i) + n_L \cdot a_n + a_n \quad (20)$$

Since the target network is identical to the main network, the total parameters for DQN is:

$$P_{DQN-t} = 2P_{DQN} \quad (21)$$

Double-DQN only modifies the Q-function update mechanism without altering the network structure, so its total number of parameters is the same as DQN:

$$P_{Double-DQN-t} = P_{DQN-t} \quad (22)$$

Dueling-DQN splits the Q-function into two branches: one to compute the state value $V(s)$ and another to compute the advantage $A(s, a)$. The shared layers' parameters are:

$$P_{share} = s_n \cdot n_1 + n_1 + \sum_{i=2}^{L} (n_{i-1} \cdot n_i + n_i) \quad (23)$$

The state value branch's parameters is:

$$P_V = n_L \cdot n_v + 2 \cdot n_v + 1 \quad (24)$$

where n_v is the number of neurons in the $V(s)$ branch, and the advantage branch's parameters is:

$$P_A = n_L \cdot n_a + n_a + n_a \cdot a_n + a_n \quad (25)$$

where n_a is the number of neurons. Thus, the total number of parameters for Dueling-DQN is:

$$P_{Dueling-DQN-t} = 2 \cdot (P_{share} + P_V + P_A) \quad (26)$$

Reinforce has only one policy network, with the same number of parameters as a single DQN.

$$P_{policy-t} = P_{DQN} \quad (27)$$

Actor-Critic has a policy network and a value network. The number of parameters in the policy network is the same as that in REINFORCE, while the number of parameters in the value network is:

$$P_{AC-V} = s_n \cdot n_1 + n_1 + \sum_{i=2}^{L} (n_{i-1} \cdot n_i + n_i) + n_L + 1 \quad (28)$$

Therefore, the total parameters is:

$$P_{AC-t} = P_{AC-V} + P_{DQN} \quad (29)$$

KFDQN incorporates a knowledge system on top of DQN. Suppose the knowledge system has r_k rules; the parameters of the knowledge system is:

$$P_{ks} = r_k \cdot a_n \quad (30)$$

The total parameters for the knowledge system is:

$$P_{ks-t} = 2 \cdot (r_k \cdot a_n) \quad (31)$$

Therefore, the total parameters for KFDQN is:

$$P_{KFDQN-t} = P_{ks-t} + P_{DQN-t} \quad (32)$$

In summary, the Reinforce algorithm exhibits the lowest parameter count among the methods discussed, followed by the Actor-Critic algorithm, which has a higher parameter count due to its separate policy and value networks. DQN and Double-DQN maintain similar parameter counts due to their comparable structures, while Dueling-DQN adds complexity with its additional value and advantage branches. The proposed KFDQN method introduces a modest increase in parameter complexity compared to DQN, as the knowledge-based fuzzy system contributes only a limited number of parameters. Overall, KFDQN's computational complexity remains competitive with standard methods while offering enhanced performance.

5. Experiment and performance evaluation

The purpose of the method proposed in this paper is to leverage knowledge to enhance the learning process, thereby improving the learning efficiency, stability, and performance of reinforcement learning agents. To validate the improvements in these three aspects, two environments CartPole-v0 and MountainCar-v0 were selected from the OpenAI Gym platform. Both test environments present problems of notable complexity. Specifically, CartPole-v0 requires precise control and balance, while MountainCar-v0 necessitates the agent to master long-term planning. These two environments allow for the performance evaluation of the agent from different dimensions. Additionally, standard implementations for both environments are available in OpenAI Gym, enabling a consistent comparison of different algorithms under the same conditions. Furthermore, to assess the proposed method's performance in real-world scenarios, practical experiments were conducted using mobile robots on tasks such as Goal Reach and obstacle avoidance.

5.1. Evaluation Metrics and Baseline

- Learning efficiency is measured by how quickly an algorithm learns to solve a task; specifically, fewer episodes required to achieve desired performance indicates higher efficiency. For immediate reward tasks like CartPole-v0, efficiency is assessed by the number of episodes needed to reach a specific cumulative reward. In contrast, for long-term planning tasks like

MountainCar-v0, it is measured by the episodes required for the task to succeed a certain number of times.

- Stability in learning refers to the consistency of the learning process, characterized by smaller fluctuations in performance during training. This is intuitively shown by plotting the average reward curve, where less fluctuation indicates better stability. For tasks with a reward benchmark, such as CartPole-v0, the Mean Absolute Deviation (MAD) quantifies stability, with a smaller deviation indicating improved stability.
- Agent performance is measured by its ability to maximize cumulative rewards over time, typically assessed through total return and average return. For tasks requiring long-term planning, the task success rate is also important. Thus, this paper evaluates performance differences using total return, average return for a fixed number of episodes, and task success rate.
- In addition to the success rate, model performance in the mobile robot Goal Reach and Obstacle Avoidance tasks can be evaluated by trajectory similarity. The line connecting the target points serves as the reference trajectory, and the actual trajectory is matched to this reference using the Dynamic Time Warping (DTW) value. Smaller DTW values indicate greater similarity, reflecting better model performance.

Given the close relationship between the method proposed in this article and DQN, DQN and its improved algorithms are used as benchmarks to measure the performance and efficiency of the proposed method. In addition, the einforce and actor critic methods were used as baseline methods.

1. **Double-DQN**: This enhancement of the DQN algorithm addresses the overestimation problem in calculating the $\max_{a'} Q_{\omega^-}(s', a')$. It separates action selection from value estimation by using the current $Q_\omega(s, a)$ for action selection and the target $Q_{\omega^-}(s, a)$ for target calculation (Kim, Kim, Jung and Oh, 2022).

2. **Dueling-DQN**: This algorithm improves insights by decomposing $Q_\omega(s, a)$ into state-value and advantage functions. Specifically, it splits the action-value function into:

$$Q_{\eta,\alpha,\beta}(s, \alpha) = V_{\eta,\alpha}(s) + A_{\eta,\beta}(s, \alpha) \tag{33}$$

Here, $V_{\eta,\alpha}(s)$ is the state-value function, and $A_{\eta,\beta}(s, a)$ represents the advantage function for different actions. The parameter η is shared between V and A, while α and β are unique to each. The action-value function is obtained by summing these two components.

3. **Reinforce**: A policy gradient method that updates the policy directly based on reward signals, using Monte Carlo methods to estimate the gradient of expected returns, making it suitable for complex environments.

4. **Actor-Critic**: This approach combines policy-based and value-based methods. The actor updates the policy directly, while the critic estimates the value function to reduce the variance of the policy gradient, balancing stability and learning efficiency across various tasks.

Table 1
CartPole-v0 action space

action	definition
0	Move left
1	Move right

Table 2
CartPole-v0 state space

Num	Observation	Min	Max
0	Cart Position	-2.4	2.4
1	Cart Velocity	-inf	inf
2	Pole Angle	-41.8°	41.8°
3	Pole Angular Velocity	-inf	inf

5.2. CartPole-v0

5.2.1. Introduction to the CartPole-v0 environment

CartPole-v0 emulates an inverted pendulum mounted on a cart, with the objective of sustaining equilibrium by maneuvering the cart. The corresponding schematic diagram and pertinent parameters are depicted in Fig.3, Table 1, and Table 2, respectively. Given that the goal is to keep the pole as vertical as possible during the cart's movement, a score of 1 is awarded each time an action is executed and the pole remains vertical. The episode concludes when the total reward reaches 200 or when the cart's position and the pole's angle exceed certain thresholds.

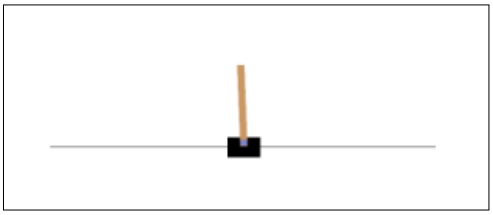


Figure 3. The CartPole-v0

5.2.2. Details of fuzzy system (CartPole-v0)

As per Table 2, the inputs to the fuzzy system are determined to be the state-space variables of the environment. For each variable, two Gaussian fuzzy sets are designed, semantically corresponding to "left" and "right." The fuzzy set associated with each input variable is depicted in Fig.4, where c_p represents the cart's position, c_v is the cart's velocity, p_d is the pole's angle, and p_v is the pole's velocity. Furthermore, Gaussian fuzzy sets are also constructed for the action space, semantically corresponding to "large" and "small," as illustrated in Fig.5.

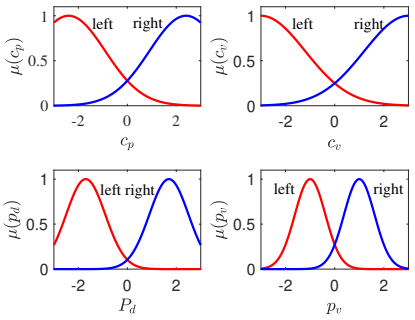


Figure 4. State fuzzy set (CartPole-v0)

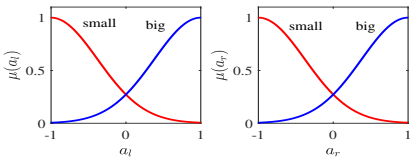


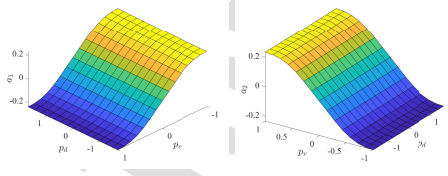
Figure 5. Action fuzzy set (CartPole-v0)

Table 3
Knowledge rule (CartPole-v0)

Num	Semantic knowledge
0	When the pendulum is to the left of the center and leans left, the cart should move left.
1	When the pendulum is to the left of the center and leans right, the cart should move right.
2	When the pendulum is to the right of the center and leans right, the cart should move right.
3	When the pendulum is to the right of the center and leans left, the cart should move left.

Based on the rudimentary human understanding of the inverted pendulum motion model, several pieces of semantic knowledge can be distilled as Table 3.

Based on this semantic knowledge, 16 corresponding fuzzy rules have been formulated, with a subset of these rules depicted in the Fig.6



(a) Rule surface for p_d, p_v, a_1 (b) Rule surface for p_d, p_v, a_2

Figure 6. Fuzzy rule (CartPole-v0)

Table 4
Parameter Setting

Items	Value
$Q(s,a)$	Input:4 hidden_dim:128 output:2
optimizer	Adam lr:0.002
γ	0.98
δ	0.01
EP	500
Buffer_size	10000
minimal_size	500
batch_size	64

5.2.3. Parameter setting

To ensure a fair comparison, aside from the parameters unique to the proposed method, all other parameters are kept consistent with the baseline method. The configuration of each parameter is presented in Table 4.

In Furthermore, the unique parameters h_1 and h_2 of the method proposed in this article are set to 0.1 and 0.08, respectively For m and n, considering that the influence of learning process parameters on fuzzy systems may need to change, a dynamic parameter that decreases over time is defined, as shown in (34).

$$m = 0.35 + 0.6e^{-i} \tag{34}$$

5.2.4. Analysis of training results (CartPole-v0)

To validate the learning efficiency of the proposed method, we set a benchmark of an accumulated reward of 50,000. We recorded the number of training epochs required to achieve the reward value using six different methods, as shown in the Fig.7. From the figure, it can be seen that the KFDQN requires the fewest episodes, indicating a higher learning efficiency.

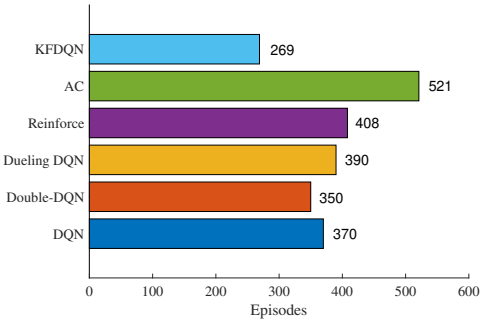


Figure 7. Efficiency comparison (CartPole-v0)

To verify the stability of the proposed method, we recorded the average reward during training with a boundary of 50 episodes, and calculated the MAD of the average reward curve based on a benchmark of 200.

In the Fig.8, the average reward curve of DQN is at a lower level before 100 episodes. This is because the parameters of the Q network are random in the early stage and need to be explored randomly, resulting in a slower learning speed. After 100 episodes, it converges to the maximum, and the subsequent round reward curve has a large fluctuation. This is because when DQN learns new states, it uses random exploration, which leads to randomness in state changes, and there is uncertainty in the model when updating the target value with non-optimal $Q_{\omega^-}(s', a')$. In contrast, Double-DQN also has a period of low reward value before 100 episodes, which is also due to poor initial network parameters and the need for random exploration. However, before 100 episodes, the reward curve of Double-DQN converges to the maximum value, indicating that Double-DQN converges faster than DQN. However, there is also a large fluctuation in the training reward after 100 episodes, which is also due to random exploration and model uncertainty. Similarly, Dueling-DQN exhibits similar behavior to the previous two methods.

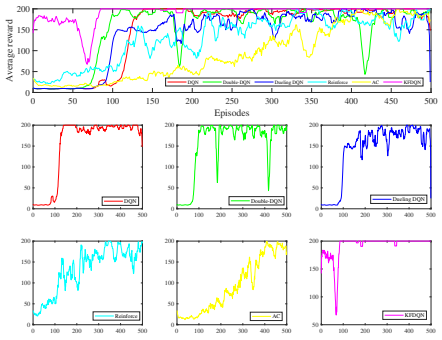


Figure 8. Average reward

In addition, compared to the previous methods, the Reinforce algorithm requires more episodes to reach a score of 150 or above. This is because the algorithm is an on-policy algorithm, and the trajectory data collected previously will not be reused. In addition, the performance of the algorithm fluctuates significantly because the residual value of each sampled trajectory fluctuates greatly. The AC algorithm uses more trajectory sequences compared to reinforcement, but the training process is more stable and has less jitter, indicating that the introduction of the value function reduces the estimation variance of the policy gradient.

In contrast, in the method proposed in this paper, the Q network undergoes supervised learning under the guidance of the fuzzy system for the first 50 episodes. Due to the inherent limitations of this knowledge, the reward function experiences fluctuations during these initial 50 episodes. Furthermore, the average reward curve reaches the maximum value earlier compared to other methods. Under the influence of mixed action selection strategies and mixed learning methods, it maintains stability after reaching the peak, with minor fluctuations in the curve. This indicates that the proposed method possesses superior stability and long-term performance.

Fig.9 shows the Mean Absolute Deviation (MAD) of the average reward curve based on a benchmark of 200 points. The MAD of the proposed method is 0.4813, which is far less than the maximum value of 89.1275 and less than other baseline methods, indicating that the stability of the proposed method is better than the baseline methods. In addition, the large MAD of AC and Reinforce algorithms is due to their slow convergence speed. Among 500 episodes, only a few rounds are close to 200, resulting in a large calculated MAD.

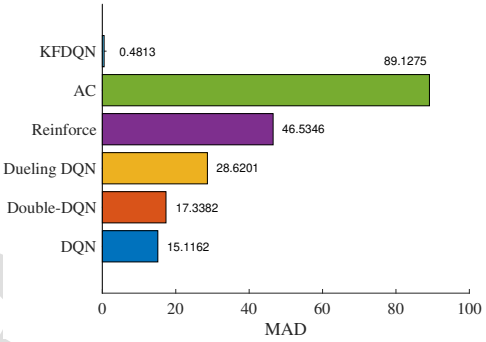


Figure 9. MAD

In order to validate the performance of the method proposed, statistics were gathered on the total return and reward segment ratio over 500 training episodes. A reward of 200 was used as the benchmark for task success, and the success rate was calculated accordingly.

Fig.10 compares the total rewards of all episodes, where the total score of the proposed method is the highest, indicating that it has the best overall performance in multi-episodes training.

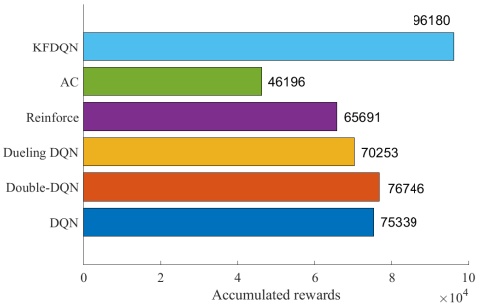


Figure 10. Total rewards

Fig.11 presents the score proportions for all episodes, where the proposed method has the highest proportion of episodes scoring 200, at 87%, and the proportion of episodes scoring above 100 is as high as 98%, which is 20% higher than the second-ranked DQN. Additionally, in terms of task success rate, the proposed method succeeds at a rate of 87%, far exceeding the second-ranked DQN. In summary, the performance of the proposed method far surpasses that of other baseline methods.

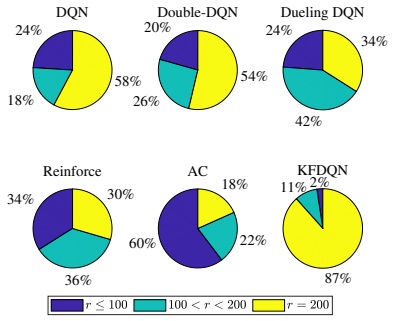


Figure 11. Reward distribution

5.2.5. *Comparison of ablation simulation (CartPole-v0)*

In order to verify the role of the HYAS in the proposed method, an ablation simulation was conducted, removing HYAS from the proposed method. The resulting cumulative reward curve is shown in Fig.12. The figure shows that after removing HYAS, the convergence time of the cumulative reward curve is significantly slower than KFDQN and DQN. This is because after the removal of the mixed action selection strategy, the guidance of knowledge is lost, leading to a longer exploration time. At the same time, due to the role of HYL in suppressing model uncertainty, the learning speed is relatively slow. In summary, this indicates that HYAS can effectively reduce the uncertainty problem brought about by random exploration, and accelerate the convergence of the model.

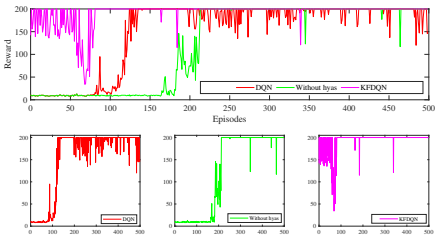


Figure 12. Reward without HYAS

Similarly, to verify the role of HYL in the proposed method, HYL was removed, and the resulting cumulative reward curve is shown in Fig.13. Due to the lack of guidance from the fuzzy system, the performance of the first 100 episodes is similar to DQN. In addition, the cumulative reward curve increases in fluctuation during the learning process to reach the maximum value, but it is less dense compared to DQN, which is due to HYAS reducing the uncertainty of random exploration. Compared to KFDQN, after removing HYL, the performance significantly decreases. In summary, this indicates that HYL can effectively reduce the uncertainty brought by the model.

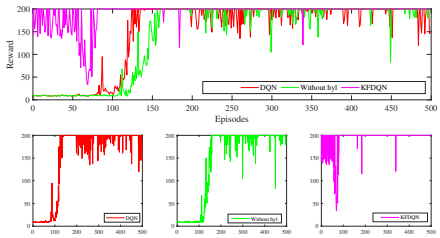


Figure 13. Reward without HYL

5.3. MountainCar-v0

5.3.1. *Introduction to the MountainCar-v0 environment*

MountainCar-v0 simulates a car on a one-dimensional track between two mountain peaks and a valley. The goal is to reach the flagpole at the right peak, but the car lacks sufficient kinetic energy to ascend directly, requiring back-and-forth movement to build speed. The schematic diagram and parameters are shown in Fig.14, Table 5, and Table 6. The agent aims to reach the target in as few steps as possible. A score of -1 is given for each time step, and the episode ends when the car either reaches the target or fails to do so within 200 steps.

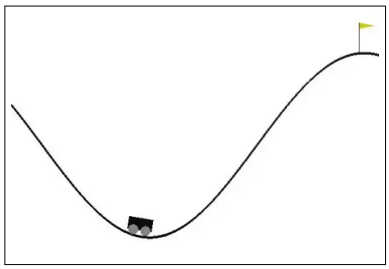


Figure 14. MountainCar-v0

5.3.2. *Details of fuzzy system (MountainCar-v0)*

According to Table 5 and Table 6, the inputs to the fuzzy system are determined to be the car's position and

Table 5
Action space (MountainCar-v0)

action	Observation
0	Accelerate to the left
1	Don't accelerate
2	Accelerate to the right

Table 6
State space (MountainCar-v0)

Num	Observation	Min	Max
0	position of the car along the x-axis	-1.2	0.6
1	velocity of the car	-0.07	0.07

velocity. For the car's position, two Gaussian fuzzy sets are established, corresponding semantically to "left" and "right". For the car's velocity, three fuzzy sets are established, corresponding semantically to "left", "stop", and "right". The fuzzy set associated with each input variable is depicted in Fig.15. Where, c_p represents the car's position, and c_v is the car's velocity.

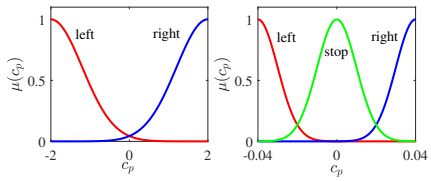


Figure 15. State fuzzy set (MountainCar-v0)

Additionally, the fuzzy set of the action space is shown in Fig.16. Based on the naive human experience of this model, several pieces of semantic knowledge can be summarized as Table.7.

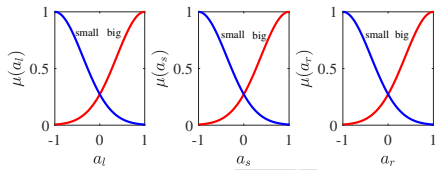
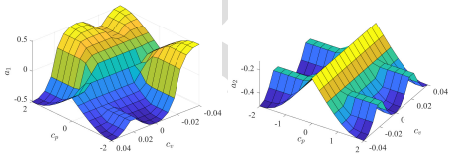


Figure 16. Action fuzzy set (MountainCar-v0)

Based on the aforementioned semantic knowledge, six corresponding fuzzy rules have been formulated. A subset of these derived fuzzy rules is depicted in Fig.17.

Table 7
Knowledge rule (MountainCar-v0)

Num	Semantic knowledge
0	When the car is on the left side of the valley and moving left, apply left thrust.
1	When the car is on the left side of the valley and moving right, apply right thrust.
2	When the car is on the left side of the valley and stopped, apply right thrust.
3	When the car is on the right side of the valley and moving right, apply right thrust.
4	When the car is on the right side of the valley and moving left, apply left thrust.
5	When the car is on the right side of the valley and stopped, apply left thrust.



(a) Rule surface for c_v, c_p, a_1 (b) Rule surface for c_v, c_p, a_2

Figure 17. Fuzzy rule (MountainCar-v0)

5.3.3. *Analysis of training results (MountainCar-v0)*

In order to validate the learning efficiency of the proposed method in long-term planning tasks, training was conducted in the MountainCar-v0 environment. Due to a greater trust in experiential knowledge, parameters h_1, h_2, m, n were changed to $[0.4, 0.6, 0.8, 0.2]$. Unlike CartPole-v0, since a reward of -1 is received for each step taken and the ultimate goal is to reach the target point, the number of training episodes required by four methods was recorded, using the completion of 250 tasks as a benchmark. Fewer episodes indicate higher efficiency. The results are shown in Fig.18.

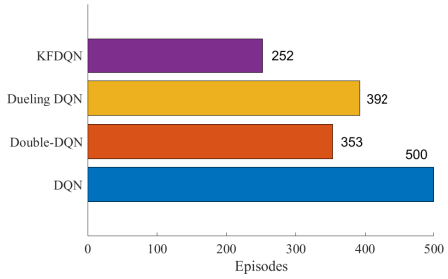


Figure 18. Efficiency comparison (MountainCar-v0)

Due to the short step count and long-term planning nature of this task, it is not conducive to collecting effective trajectories. Under the same parameters, both the Reinforce and AC methods failed to converge during the 500 episodes,

so their results were not presented. From the Fig.18, it can be seen that the proposed method requires 252 episodes to complete 250 tasks, which is less than the other baseline methods, indicating higher learning efficiency.

To validate the stability of the proposed method, the average reward during training was recorded over 50 episodes, as shown in the Fig.19. Since MountainCar-v0 is a long-term planning task and the steps to complete the task vary in different states, it is impossible to find a benchmark for the reward, hence the Mean Absolute Deviation (MAD) cannot be calculated. From the figure, it can be observed that the reward curve of the proposed method remains relatively stable from the beginning to the end of the training. After normalizing the curve, the variance was calculated as KFDQN(0.0489), Double DQN (0.059), and Dueling DQN (0.0805). The variance results show that the fluctuation of the proposed method is smaller, indicating that the proposed method has relatively good stability.

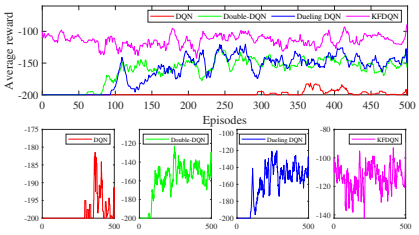
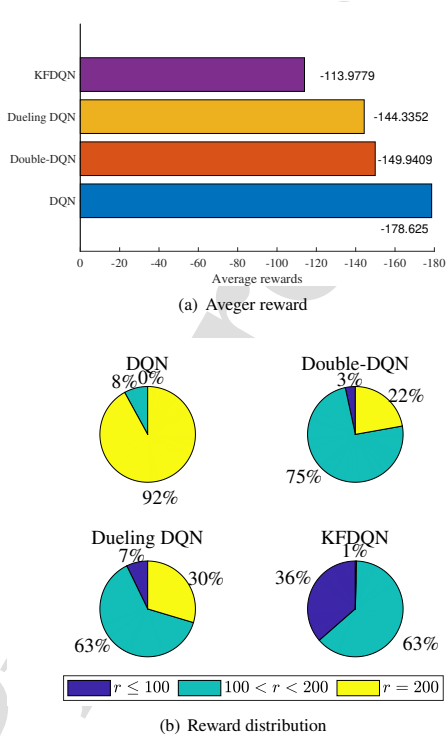


Figure 19. Aveger reward (MountainCar-v0)

To verify the performance of the proposed method, statistics were collected on the average return and reward segment ratio of successful episodes during the training process. From this, the success rate of the task was obtained, and the results are shown in Fig.20.

Fig.20(a) records the average reward for all successful episodes. From the figure, it can be seen that the average value of KFDQN is -113.9779, while the other three baseline methods are -114.3352, -149.9409, and -178.625 respectively. Compared with these, the average value of KFDQN is the lowest, indicating that the average performance of KFDQN in successful episodes is superior to other methods.

Fig.20(b) presents the score proportions for all episodes. The yellow section, where the score equals -200, represents unsuccessful episodes, while the other two sections represent successful episodes with achieved goals. A higher score indicates that the car reached the target point with fewer steps, implying better performance. As can be seen from the figure, the proportion of KFDQN scores less than -100 is the largest among all baseline methods, at 36%, and the proportion of scores above -200 reaches as high as 99%. This suggests that under the influence of KFDQN, the success rate of achieving the goal is as high as 99%. The proportions of Double DQN and Dueling DQN scores above -100 are 3% and 7% respectively, indicating that their



(a) Aveger reward

(b) Reward distribution

Figure 20. Performance comparison (MountainCar-v0)

high-score performances are far lower than that of KFDQN. Furthermore, in terms of success rate, Double DQN and Dueling DQN are 78% and 70% respectively, both lower than KFDQN, indicating that the proposed method significantly outperforms other baseline methods.

5.3.4. Comparison of ablation simulation (MountainCar-v0)

In order to analyze the role of the HYAS, the strategy is similarly removed from the proposed method. The resulting cumulative reward curve is shown in Fig.21.

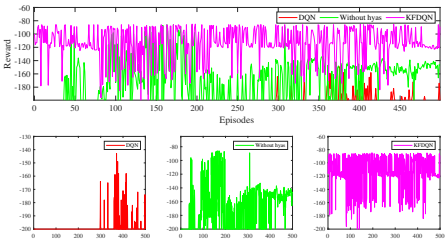


Figure 21. Reward without HYAS (MountainCar-v0)

The figure shows that after removing the HYAS, the cumulative reward curve is at a lower reward for the first 50 episodes. However, it converges much earlier compared to DQN, indicating that even after removing the mixed action selection strategy, the mixed learning strategy still aids the learning of the agent. In addition, compared to KFDQN, the performance of 'Without hyas' shows a declining trend after 200 episodes and an upward trend after 250 episodes, with noticeable fluctuations. In contrast, the reward curve of KFDQN is more stable, suggesting that the HYL has a positive effect on the stability of the learning process. At the same time, KFDQN converges faster, indicating that HYAS plays a certain role in accelerating the learning of the agent.

Similarly, in order to verify the role of the mixed learning method in the proposed method, the mixed learning method is removed. The resulting cumulative reward curve is shown in Fig.22.

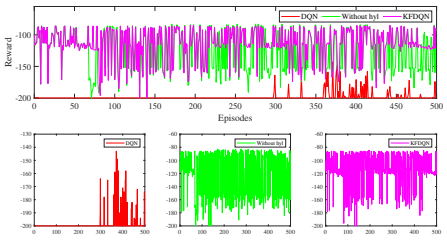


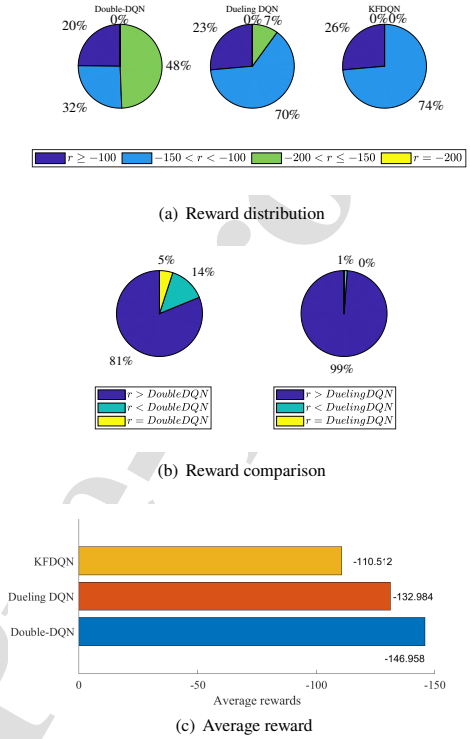
Figure 22. Reward without HYL (MountainCar-v0)

As can be seen from the figure, after removing HYL, the convergence speed is significantly ahead of DQN, which verifies the role of the mixed action selection strategy in accelerating the learning of the agent. However, after 100 episodes, the reward curve fluctuates between -90 and -160. In comparison, KFDQN also has fluctuations, but the density is relatively small, indicating that the lack of hyl has a negative impact on the stability of the agent's learning. In summary, this suggests that the HYL can effectively reduce the uncertainty brought by the model.

5.4. Analysis of simulation test results

To validate the performance of the trained model, tests were conducted in the aforementioned two environments. In the cartpole-v0 environment, all four methods scored 200 points in all test episodes, showing no significant differences. Therefore, only the test results in the MountainCar environment are analyzed. To obtain a state different from the training, the random seed for state generation was changed to 1. After 500 episodes of testing, the results are Fig.23.

Due to the poor convergence of DQN during training, resulting in a pass rate of 0 in the test, it is not discussed here. Fig.23(a) counts the score proportions of all episodes of the three methods, and analyzes the differences in test performance of the three methods through the proportions of different segments. The yellow part, i.e., the score equals -200,



(a) Reward distribution

(b) Reward comparison

(c) Average reward

Figure 23. Test results analysis (MountainCar-v0)

belongs to the unsuccessful part, and the other three parts are the scores of the episodes where the goal is achieved. The higher score, the fewer steps the car takes to reach the target point, indicating that the car performs better. From the figure, it can be seen that the proportion of KFDQN less than -100 points is larger than that of other baseline methods, accounting for 26%, the proportion of higher than -150 and less than -100 is 74%, and the proportion of more than -200 reaches 100%, indicating that the success rate of achieving the goal under the action of KFDQN is as high as 100%. The proportions of Double DQN and Dueling DQN above less than 100 points are 20% and 23% respectively, and their high scores are lower than KFDQN. In addition, in the segment of $-200 < r < 150$, the proportions of Dueling DQN and Double DQN are 7% and 48%, both higher than KFDQN (0%). This indicates that the performance of the proposed method is at a relatively high level, superior to other baseline methods.

Fig.23(b) records the comparison of the reward scores of KFDQN, Double DQN, and Dueling DQN in each round, measuring the performance difference of KFDQN and the baseline method in a single round. From the figure, it can be seen that in all single episodes, the proportion of KFDQN

greater than the reward of Double DQN is 81%, and the proportion of greater than the reward of Dueling DQN is 99%, indicating that from the perspective of a single round, the performance of the proposed method is far superior to other baseline methods.

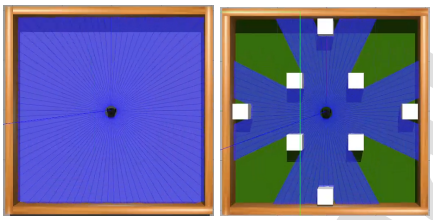
Fig.23(c) records the average reward value of all successful episodes. From the figure, it can be seen that the average value of KFDQN is -110.512, and the other baseline methods are -132.984 and -146.958 respectively. Compared with them, the average value of KFDQN is the lowest, indicating that the average performance of KFDQN's successful episodes is superior to other methods.

In summary, from the three perspectives of different segment performance, single round performance, and average performance, the proposed method is superior to other baseline methods, verifying the effectiveness of the proposed method.

5.5. Goal reach and obstacle avoidance

5.5.1. Environmental Introduction

The training environments for the mobile robot tasks, specifically goal reaching and obstacle avoidance, are illustrated in the Fig.24. The goal reaching task is conducted in an obstacle-free environment, where the mobile robot is required to navigate to a specified target point. Fig.24(b) depicts the obstacle avoidance task, in which the mobile robot must maneuver around obstacles to reach the designated target point. The MDP settings for these tasks are defined as follows:



(a) Env1 (Goal Reach) (b) Env2 (Obstacle Avoidance)

Figure 24. Experimental training environment

- **State Space:** The state space consists of 90-dimensional laser radar point cloud data, 2-dimensional target position represented by (θ_d, dis), and a 1-dimensional representation of the previous action taken.Where θ_d is the angle with the target, and dis is the distance from the target point.
- **Action Space:** The action space includes three possible actions: left turn, right turn, and move forward.
- **Reward Function:** The reward function is designed as (35), which evaluates the robot's performance based on its proximity to the target and the avoidance of obstacles.

$$\begin{cases} r_{reach} if\ d_t < RTH \\ r_{collision} if\ min_{x_i} < CTH \\ (d_{t-1} - d_t)p_r \\ r_o \end{cases} \tag{35}$$

where d_t is the distance to the target point, $\min_{x_i}$ is the minimum value of the LiDAR point cloud, p_r is a proportional parameter, and r_o is the reward for each step taken. RTH and CTH are the reaching threshold and collision threshold, respectively.

5.5.2. Network structure and parameter settings

The network structures corresponding to different methods are shown in Fig.25. The parameter settings are shown in the Table.8. The knowledge is shown in Table.9 and Table.10.

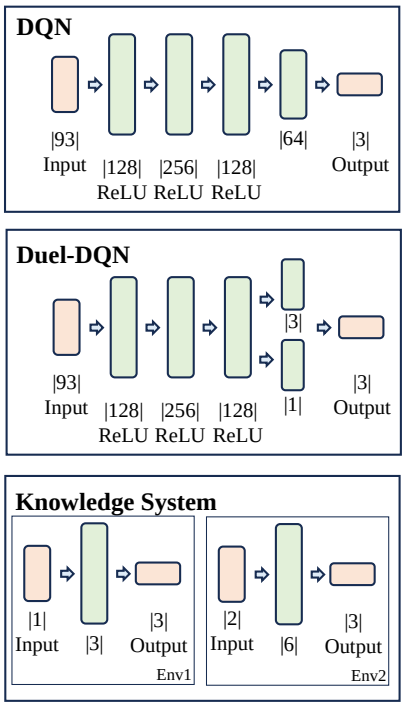


Figure 25. Network structure

5.5.3. Analysis of training results (Goal Reach and obstacle avoidance)

Due to the fact that both the Reinforce and Actor-Critic methods failed to converge under the same parameters and training episodes, their training results are not presented here. Additionally, since the goal of both the Goal Reach and Obstacle Avoidance tasks is to reach a specified target, the

Table 8
Parameter Setting (Goal Reach and Obstacle Avoidance)

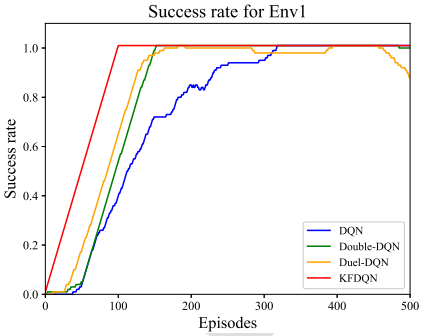
Items	Value
optimizer	Adam lr:0.0001
γ	0.99
δ	0.01
EP	500(Env1) 1000(Env2)
Buffer_size	10000
minimal_size	1500
batch_size	256

Table 9
Knowledge rule (Env1)

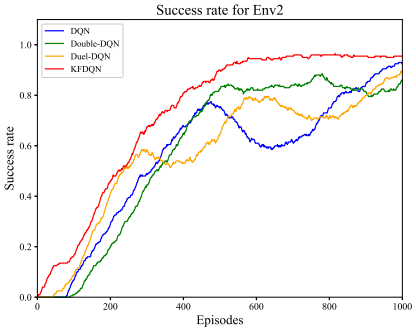
Num	Semantic knowledge
0	When the target is on the left, turn left
1	When the target is on the right, turn right
2	When the target is if front, move forword

Table 10
Knowledge rule (Env2)

Num	Semantic knowledge
0	Close to obstacles with the target right, turn right
1	Close to obstacles with the target left, turn left
2	Close to obstacles with the target ahead, turn left or right.
3	Far from obstacles with the target left, turn left.
4	Far from obstacles with the target right, turn right.
5	Far from obstacles with the target ahead, move forword



(a) Success rate for Env1



(b) Success rate for Env2

Figure 26. Success rate for Env1 and Env2

learning efficiency is measured by the number of training episodes required to achieve a certain task success rate. The results are shown in Fig.26. The success rate graph for Env1 demonstrates that KFDQN significantly outperforms the other methods in both learning efficiency and stability, achieving a 1.0 success rate in fewer than 100 episodes with no fluctuations. In comparison, Double-DQN and Duel-DQN converge around 300 episodes with moderate stability, while DQN is the slowest, taking over 400 episodes to reach a similar success rate, showing both lower efficiency and minor fluctuations in stability. Overall, KFDQN provides the most rapid and consistent learning performance. The success rate graph for Env1 demonstrates that KFDQN significantly outperforms the other methods in both learning efficiency and stability, achieving a 1.0 success rate in fewer than 100 episodes with no fluctuations. In comparison, Double-DQN and Duel-DQN converge around 300 episodes with moderate stability, while DQN is the slowest, taking over 400 episodes to reach a similar success rate, showing both lower efficiency and minor fluctuations in stability. Overall, KFDQN provides the most rapid and consistent learning performance.

Table 11
Run-time for success rate of 75%

	DQN	Doubel-DQN	Dueling-DQN	KFDQN
run-time(Env1)	2957.67	2174.70	2272.76	**1565.39**
run-time(Env2)	3919.78	3923.80	5413.21	**3717.81**

Table.11 presents the run-time comparisons of different algorithms in Env1 and Env2. In Env1, KFDQN demonstrates the shortest run-time, completing in 1565.39 seconds, which is significantly faster compared to DQN (2957.67 seconds), Double-DQN (2174.70 seconds), and Dueling-DQN (2272.76 seconds). Similarly, in Env2, KFDQN again achieves the best performance with a run-time of 3717.81 seconds, outperforming DQN (3919.78 seconds), Double-DQN (3923.80 seconds), and Dueling-DQN (5413.21 seconds). These results suggest that KFDQN not only improves learning efficiency but also reduces the overall computational time, making it a more efficient approach for both environments.

5.5.4. *Sim-to-real*

Goal Reach: We selected models trained for 4,000 and 6,000 steps and conducted real-world experiments in the environment shown in Fig.27, with targeting points $(0.7, 0.5)$, $(1.6, 0.5)$, $(1.3, -0.4)$, $(0.8, 0.0)$, and $(0.8, -0.5)$. Taking the 6,000-step model as an example, the real-world trajectory is depicted in Fig. 28. The complete trajectories for all models are shown in Fig.29, where Fig.29(a) shows the trajectory of the 4000-step model experiment. The following results were observed: DQN failed to reach any of the five target points, resulting in a success rate of 0%. Double-DQN successfully reached four target points, achieving an 80% success rate, while Dueling-DQN reached two points, with a success rate of 40%. In contrast, KFDQN successfully reached all target points, achieving a success rate of 100%. In addition, since only KFDQN completed all points, trajectory similarity analysis could not be performed for comparison with other methods. These results demonstrate that the proposed method can complete the task with fewer training steps, indicating higher learning efficiency.

Figure 27. Real experimental environment (Env1)

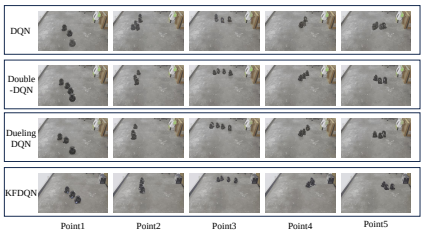
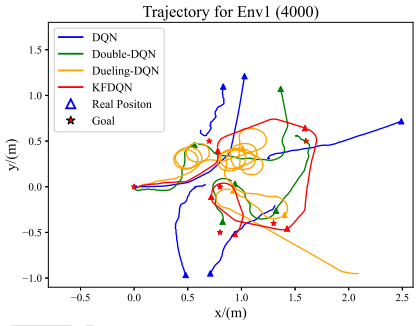


Figure 28. Real-world trajectory for Env1 (6000)

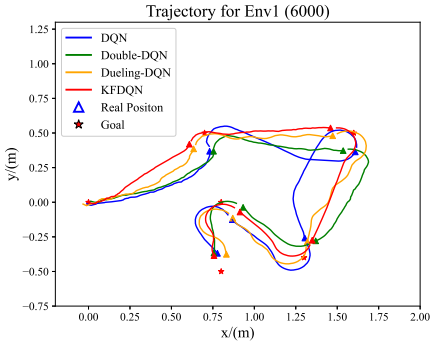
Fig.29(b) presents the trajectories for models trained with 6000 steps. In this case, all methods successfully completed all the target points, with a task success rate of 100%, showing that all methods improved after an additional 2000 steps of training. To further distinguish the methods, we calculated the DWT, as shown in Table.12. KFDQN achieved the lowest score of 56.715, indicating the highest similarity between its trajectory and the optimal shortest path connecting the target points, demonstrating that KFDQN exhibits the best performance.

Table 12
DTW (Env1)

	DQN	Doubel-DQN	Dueling-DQN	KFDQN
DTW	73.268	62.360	62.506	**56.715**



(a) Trajectory rate for Env1 (4000)



(b) Trajectory rate for Env1 (6000)

Figure 29. Trajectory for Env1

Obstacle Avoidance: In the obstacle avoidance task, models trained for 28,000 and 35,000 steps were tested in a real environment, as shown in Fig. 30. The selected target points were $(1.4, 0.9)$ and$(-0.15, 0.1)$. Taking the experiments from the 35,000-step model as an example, the trajectory in the real scene is depicted in Fig. 31, while the complete trajectory is illustrated in Fig.32. From Fig.32(a), it can be seen that all methods only completed one target point. As shown in Fig.32(b), Double DQN reached only one target, while Dueling DQN failed to reach any target. In contrast, both DQN and KFDQN successfully reached all targets, with KFDQN demonstrating shorter and smoother

trajectories compared to DQN. The DTW values for DQN and KFDQN were calculated as 256.480 and 149.468, respectively. These results indicate that KFDQN achieved better performance in a shorter training period, suggesting an improvement in learning efficiency.

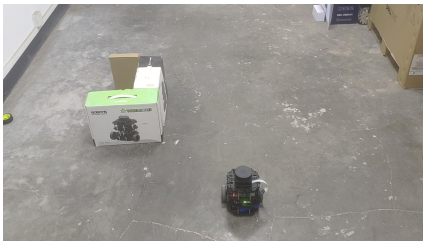


Figure 30. Real experimental environment (Env2)

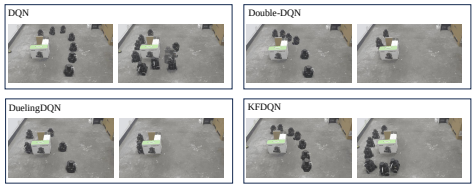
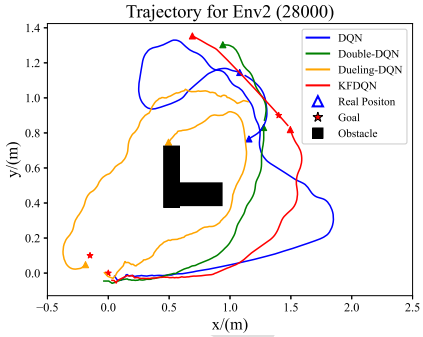
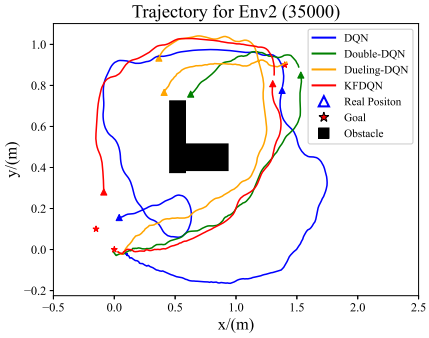


Figure 31. Real-world trajectory for Env2 (35000)



(a) Trajectory rate for Env2 (28000)



(b) Trajectory rate for Env1 (35000)

Figure 32. Trajectory for Env2

5.5.5. *Decision process analysis*

To better understand the relationship between model inputs and outputs, we conducted a decision-making process analysis using the mobile robot's goal reach task as an example. Three target points were selected: (0.4, 0.0), (0.6, 0.5), and (1.2, 0.2), which correspond to the robot's front, left front, and right front, respectively. This allowed us to analyze the relationship between different deviation angles and the model outputs. By recording the model outputs and deviation angles during the movement process, we obtained the relationship graph shown in Figure a.

Fig.33(a): For the target point (0.4, 0.0), the Q-value for moving forward (green dots) is higher compared to turning left or right, indicating that the model prefers a straight action when θ_d is near 0.

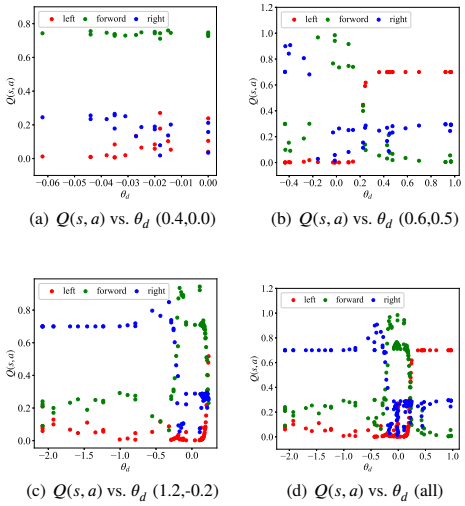
Fig.33(b): For the target point (0.6, 0.5), θ_d varies from -0.4 to 1.0. The model outputs a left turn signal when $\theta_d > 0.2$, a forward signal when θ is between -0.2 and 0.2, and a right turn signal when $\theta_d < -0.2$.

Fig.33(c): For the target point (1.2, -0.2), θ_d changes from -2 to near 0. The Q-value for turning right is higher within the θ_d range of (-2, -0.25), while the Q-value for moving forward is higher as θ_d approaches 0.

Combining the data from all figures, Fig.33(d) clearly shows that when θ_d is near 0, the Q-value for moving forward is higher. When $\theta_d < 0$, the Q-value for right turns is higher, and when $\theta_d > 0$, the Q-value for left turns is higher. Given these observations, we can clearly observe the correlation between the model's output and the deviation angle: when the target point is on the left, the model provides a left-turn signal; when the target point is on the right, the model issues a right-turn signal; and when the target point is directly ahead, the model indicates a go-straight signal.

5.6. Discussion

RL faces challenges such as low learning efficiency due to random interactions and instability caused by Q-value estimation bias. The method we propose sheds light on how human prior knowledge can be introduced to alleviate these issues. Specifically, we leverage human semantic knowledge in the form of fuzzy rules to guide the learning process of RL agents through a hybrid approach combining interaction, supervision, and reinforcement. As discussed earlier, the main advantage of our method is its ability to reduce the randomness in interactions and uncertainty in learning targets, which enhances learning efficiency, stability, and overall performance. This approach holds potential to advance research and applications in the field of knowledge-guided RL. However, some limitations of the proposed method should be acknowledged. Due to the reliance on fuzzy systems and human knowledge for knowledge representation, the



(a) $Q(s,a)$ vs. θ_d (0.4,0.0) (b) $Q(s,a)$ vs. θ_d (0.6,0.5)

(c) $Q(s,a)$ vs. θ_d (1.2,-0.2) (d) $Q(s,a)$ vs. θ_d (all)

Figure 33. $Q(s,a)$ vs. θ_d

method may not adequately represent and utilize features beyond the key dimensions. Additionally, a unified approach for integrating multi-perspective knowledge has not been achieved, and the introduction of more hyper-parameters makes tuning these parameters for optimal performance potentially time-consuming. Although interpretability is not the primary focus of our study and we acknowledge the lack of exploration in this area, we believe it is crucial for understanding how the model operates and for advancing research on uncertainty quantification. To address these limitations, our short-term future work will focus on improving the algorithm, particularly in terms of automatic rule generation and optimizing the current framework. Long-term objectives include a deeper exploration of uncertainty quantification methods, investigating how knowledge can enhance model interpretability, and expanding the application of this method to more complex environments.

6. Conclusion

This paper investigates how to leverage semantic knowledge to guide reinforcement learning, with the aim of enhancing the efficiency, performance, and stability of reinforcement learning. It primarily addresses the representation and introduction of knowledge, focusing on three key challenges: the integration of semantic knowledge representation with existing reinforcement learning frameworks, the incorporation of knowledge into action selection, and the integration of knowledge into the learning process. A framework that combines supervised learning with reinforcement learning is proposed, along with a hybrid action selection strategy, a hybrid learning method, and a knowledge updating module. Compared to traditional reinforcement learning methods, this approach uses knowledge to guide learning, thereby improving learning efficiency. Compared to the random exploration mechanism, the hybrid action selection strategy integrates knowledge, which can reduce the randomness of exploration to a certain extent. At the same time, the hybrid learning method also incorporates knowledge into the construction of the objective function, reducing the uncertainty of the objective function. In addition, knowledge updating effectively reduces the negative impact of knowledge limitations on learning. Although the effectiveness and feasibility of the proposed method have been validated through simulations and experiments, demonstrating an approximate 28.6% improvement in learning efficiency and a 19.56% enhancement in performance, it is important to acknowledge that the method still has some limitations. Knowledge representation methods are ineffective in representing features beyond the key dimensions and fail to unify the integration of multi-perspective knowledge. Additionally, the introduction of more hyper-parameters has increased the difficulty of tuning. Therefore, improving the construction of knowledge rules to accommodate a broader range of knowledge with higher dimensionality presents significant research opportunities. Future work will focus on developing improved algorithms for automatic rule generation and framework optimization, as well as exploring uncertainty quantification, interpretability issues, and integrating our approach with other parallel frameworks such as distributed reinforcement learning.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Enquiries about data availability should be directed to the authors.

Acknowledgments

This work is supported by Sichuan Science and Technology Program (2024NSFSC0445).

References

Chen, C., Zhai, Y., Gao, Z., Xu, K., Yang, S., Li, Y., Ding, B., Feng, D., Wang, H., 2023. Nuclear norm maximization based curiosity-driven reinforcement learning. IEEE Transactions on Artificial Intelligence .

Chen, S., Qiu, X., Tan, X., Fang, Z., Jin, Y., 2022. A model-based hybrid soft actor-critic deep reinforcement learning algorithm for optimal ventilator settings. Information sciences 611, 47–64.

Chiou, S.W., 2024. A knowledge-assisted reinforcement learning optimization for road network design problems under uncertainty. Knowledge-Based Systems 292, 111614.

Du, Y., Chen, J., Zhao, C., Liao, F., Zhu, M., 2023. A hierarchical framework for improving ride comfort of autonomous vehicles via deep reinforcement learning with external knowledge. Computer-Aided Civil and Infrastructure Engineering 38, 1059–1078.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al., 2018. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures, in: International conference on machine learning, PMLR. pp. 1407–1416.

Eßer, J., Bach, N., Jestel, C., Urbann, O., Kerner, S., 2023. Guided reinforcement learning: A review and evaluation for efficient and effective real-world robotics [survey]. IEEE Robotics & Automation Magazine 30, 67–85.

Gao, X., Si, J., Huang, H., 2023. Reinforcement learning control with knowledge shaping. IEEE Transactions on Neural Networks and Learning Systems .

Gu, X., 2023. A dual-model semi-supervised self-organizing fuzzy inference system for data stream classification. Applied Soft Computing 136, 110053.

Guan, Y., Ren, Y., Li, S.E., Sun, Q., Luo, L., Li, K., 2020. Centralized cooperation for connected and automated vehicles at intersections by proximal policy optimization. IEEE Transactions on Vehicular Technology 69, 12597–12608.

Kaufmann, E., Bauersfeld, L., Loquercio, A., Müller, M., Koltun, V., Scaramuzza, D., 2023. Champion-level drone racing using deep reinforcement learning. Nature 620, 982–987.

Kim, M., Kim, J., Jung, M., Oh, H., 2022. Towards monocular vision-based autonomous flight through deep reinforcement learning. Expert Systems with Applications 198, 116742.

Kiran, B.R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A.A.A., Yogamani, S., Pérez, P., 2022. Deep reinforcement learning for autonomous driving: A survey. IEEE Transactions on Intelligent Transportation Systems 23, 4909–4926.

Li, S.E., 2023. Deep reinforcement learning, in: Reinforcement learning for sequential decision and optimal control. Springer, pp. 365–402.

Li, X., Wang, X., Zheng, X., Jin, J., Huang, Y., Zhang, J.J., Wang, F.Y., 2022. Sadrl: Merging human experience with machine intelligence via supervised assisted deep reinforcement learning. Neurocomputing 467, 300–309.

Lin, H., He, Y., Li, F., Liu, Q., Wang, B., Zhu, F., 2024. Taking complementary advantages: Improving exploration via double self-imitation learning in procedurally-generated environments. Expert Systems with Applications 238, 122145.

Mei, Z., Zhao, T., Xie, X., 2024. Hierarchical fuzzy regression tree: A new gradient boosting approach to design a tsk fuzzy model. Information Sciences 652, 119740.

Mendel, J.M., 2017. Uncertain rule-based fuzzy systems. Introduction and new directions 684.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al., 2015. Human-level control through deep reinforcement learning. nature 518, 529–533.

Pitombeira-Neto, A.R., Santos, H.P., da Silva, T.L.C., de Macedo, J.A.F., 2024. Trajectory modeling via random utility inverse reinforcement learning. Information Sciences 660, 120128.

Song, L., Li, D., Xu, X., 2022. Sparse online maximum entropy inverse reinforcement learning via proximal optimization and truncated gradient. Knowledge-Based Systems 252, 109443.

Sun, Q., Fang, J., Zheng, W.X., Tang, Y., 2022. Aggressive quadrotor flight using curiosity-driven reinforcement learning. IEEE Transactions on Industrial Electronics 69, 13838–13848.

Wang, L., Fernandez, C., Stiller, C., 2022. High-level decision making for automated highway driving via behavior cloning. IEEE Transactions on Intelligent Vehicles 8, 923–935.

Wang, Y., Chen, Z., Sun, M., Sun, Q., 2024. Enhancing active disturbance rejection design via deep reinforcement learning and its application to autonomous vehicle. Expert Systems with Applications 239, 122433.

Wei, D., Zhou, J., Zhu, Y., Ma, J., Ma, S., 2023. Axis-space framework for cable-driven soft continuum robot control via reinforcement learning. Communications Engineering 2, 61.

Wu, J., Huang, Z., Lv, C., 2022. Uncertainty-aware model-based reinforcement learning: Methodology and application in autonomous driving. IEEE Transactions on Intelligent Vehicles 8, 194–203.

Xu, D., Zhu, F., Liu, Q., Zhao, P., 2021. Improving exploration efficiency of deep reinforcement learning through samples produced by generative model. Expert Systems with Applications 185, 115680.

Yala, A., Mikhael, P.G., Lehman, C., Lin, G., Strand, F., Wan, Y.L., Hughes, K., Satuluru, S., Kim, T., Banerjee, I., et al., 2022. Optimizing risk-based breast cancer screening policies with reinforcement learning. Nature medicine 28, 136–143.

Yang, Z., Ren, K., Luo, X., Liu, M., Liu, W., Bian, J., Zhang, W., Li, D., 2022. Towards applicable reinforcement learning: Improving the generalization and sample efficiency with policy ensemble. arXiv preprint arXiv:2205.09284 .

Zadeh, L.A., 1965. Fuzzy sets. Information and control 8, 338–353.

Zadeh, L.A., 1992. Knowledge representation in fuzzy logic, in: An introduction to fuzzy logic applications in intelligent systems. Springer, pp. 1–25.

Zhang, Z., Wu, Z., Zhang, H., Wang, J., 2022. Meta-learning-based deep reinforcement learning for multiobjective optimization problems. IEEE Transactions on Neural Networks and Learning Systems .

Zhao, F., Wang, Q., Wang, L., 2023. An inverse reinforcement learning framework with the q-learning mechanism for the metaheuristic algorithm. Knowledge-Based Systems 265, 110368.

Zhao, T., Qin, P., Dian, S., Guo, B., 2024. Fractional order sliding mode control for an omni-directional mobile robot based on self-organizing interval type-2 fuzzy neural network. Information Sciences 654, 119819.

Zhu, Z., Lin, K., Jain, A.K., Zhou, J., 2023. Transfer learning in deep reinforcement learning: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence .

ORCID INFORMATION

https://orcid.org/0000-0001-5127-5301 (Tao Zhao)
https://orcid.org/0009-0007-9333-3478 (Peng Qin)

Highlights

- Fuzzy systems, knowledge updates, and knowledge guidance have been integrated.
- A hybrid action selection strategy is proposed to quickly obtain effective data.
- A hybrid learning method is proposed to reduce the uncertainty.

Credit Author Statement

[Peng Qin]: Conceptualization, Methodology, Software, Writing - Original Draft, Writing - Review & Editing, Visualization.

[Tao Zhao]: Conceptualization, Data Curation, Investigation, Supervision, Project Administration, Funding Acquisition.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: