Journal of Marine Science and Application (2025) 24: 1239-1251
https://doi.org/10.1007/s11804-025-00652-1

RESEARCH ARTICLE

Deep Reinforcement Learning Approach for X-rudder AUVs Fault Diagnosis Based on Deep Q-network

Chuanfa Chen[1], Xiang Gao[2], Yueming Li[1], Xuezhi Chen[1], Jian Cao[1] and Yinghao Zhang[3]

Received: 12 July 2024 / Accepted: 11 October 2024

Abstract

The rudder mechanism of the X-rudder autonomous underwater cehicle (AUV) is relatively complex, and fault diagnosis capability is an important guarantee for its task execution in complex underwater environments. However, traditional fault diagnosis methods currently rely on prior knowledge and expert experience, and lack accuracy. In order to improve the autonomy and accuracy of fault diagnosis methods, and overcome the shortcomings of traditional algorithms, this paper proposes an X-steering AUV fault diagnosis model based on the deep reinforcement learning deep Q network (DQN) algorithm, which can learn the relationship between state data and fault types, map raw residual data to corresponding fault patterns, and achieve end-to-end mapping. In addition, to solve the problem of few X-steering fault sample data, Dropout technology is introduced during the model training phase to improve the performance of the DQN algorithm. Experimental results show that the proposed model has improved the convergence speed and comprehensive performance indicators compared to the unimproved DQN algorithm, with precision, recall, $F_{1-score}$, and accuracy reaching up to 100%, 98.07%, 99.02%, and 98.50% respectively, and the model's accuracy is higher than other machine learning algorithms like back propagation, support vector machine.

Keywords Autonomous underwater cehicles; X-rudder; Fault diagnosis; Deep Q network; Dropout technique

1 Introduction

Autonomous underwater cehicles (AUVs) are widely used in marine environment observation, resource investigation, security, and defense due to their strong autonomy, wide detection range, diverse detection means, and strong concealment (Huang et al., 2020). X-rudder AUVs are more efficient and maneuverable in performing operational tasks than cross-rudder AUVs. Thus, their applications are more promising. However, the four rudder blades are independent of one another, the control allocation is complicated, and the rudder is prone to jamming, winding, deformation, fracture, and other faults due to the complex marine environment (Yuan et al., 2023). Thus, AUVs must be capable of automatic fault diagnosis, which is conducive to the timely adoption of targeted measures to control movement under fault conditions while ensuring the safety of carriers to continue operational tasks or emergency surfacing.

Traditional fault diagnosis methods can be categorized as analytical model-based, signal processing-based, and knowledge-based (Frank, 1990). The analytical model-based approach has been the earliest proposed diagnostic method. It models the object to be diagnosed and performs fault diagnosis by comparing the estimated state output from the ideal model with the state obtained from actual measurements. Alessandri (2003) used a feed-forward neural network to fit the dynamics model of AUVs for establishing a fault observer. It involves comparing the residuals between the observer state output and the measured state values for fault diagnosis. Wang et al. (2007) proposed a fault fusion diagnosis method for AUV thrusters, which fuses overall and local fault information for fault diagnosis. Sun et al. (2016) proposed an improved Gaussian particle filtering algorithm for fault modeling and motion state estimation of AUVs to realize fault diagnosis for

Article Highlights

- "Troubleshooting Quiz Game" model is developed to transform supervised learning problems into reinforcement learning problems.
- A new fault diagnosis method based on deep reinforcement learning has been proposed, which can achieve the mapping of raw residual data to fault patterns.
- Improve the DQN algorithm by combining with the Dropout technique, which overcome the problem of overfitting triggered by small sample size.

✉ Yueming Li
liyueming@hrbeu.edu.cn

[1] National Key Laboratory of Autonomous Marine Vehicle Technology, Harbin Engineering University, Harbin 150001, China

[2] Jiangsu Automation Research Institute, Lianyungang 222061, China

[3] Wuhan Second Ship Design and Research institute, Wuhan 430000, China

AUV motion actuators. However, given that AUVs have multiple degrees of freedom and are strongly coupled nonlinear systems, obtaining an accurate mathematical model is difficult, which can affect the diagnostic results.

Signal processing-based fault diagnosis methods extract eigenvalues such as amplitude, frequency, and phase from directly measurable input and output information and their trends. These eigenvalues are then used for fault diagnosis. Zhang et al. (2015) proposed a fault feature extraction method based on wavelet approximation components and control signal rate of change. A feature fusion method with normalization was employed to improve the accuracy of propeller fault diagnosis under stochastic disturbances. Miskovic and Barisic (2005) utilized a probabilistic principal component analysis technique for fault diagnosis of underwater robot actuators with sensors. Similarly, the measurement signal of AUVs is noisy due to interference from environmental conditions, among other factors, which leads to limitations in the application of the method.

Knowledge-based fault diagnosis methodology depends on a priori knowledge of the operation mechanism of the system being diagnosed, fault characteristics, and response to faults, which leads to fault diagnosis. Wang et al. (2011) established a qualitative model of the dynamics of the "Beaver" with a qualitative diagnostic model of thruster failure. Liu and Xu (2016) addressed the fault localization problem of the rudder of a torpedo-shaped AUV. They established a table of positive and negative properties of the rudder deformation fault factor through qualitative force analysis. The rudder deformation fault was isolated by combining the fault detection results. However, this method relies heavily on human experience and expert knowledge.

In recent years, data-driven fault diagnosis methods have been widely researched. At their core, these methods utilize machine learning to learn fault characteristics from data and construct fault diagnosis models with more objectivity and accuracy (Jiang et al., 2018; Jiang and Yin, 2019; Ji et al., 2021). Machine learning algorithms analyze vast amounts of data to identify patterns and trends within the data, which they then use to predict outcomes for new data. Their ability to learn autonomously and continuously makes them highly suitable for fault diagnosis. For example, Zhao et al. (2024) developed a lightweight deep learning model called ShuffleNetv2 and Coordinate Attention Single Shot Multibox Detection (SN-CA-SSD) to detect defects in turbine blades, which improves detection accuracy and efficiency. Žarković and Stojković (2017) presented an artificial intelligence-based methodology utilizing fuzzy logic to enhance fault detection and classification of power transformers, which assists operators in making informed decisions regarding maintenance and intervention urgency. Alex et al. (2020) implemented machine learning algorithms for fault classification in vehicle power transmission systems. These algorithms use acoustic signals to diagnose faults with high accuracy. Machine learning algorithms are typically categorized into four types according to their learning paradigms: supervised learning, unsupervised learning, weakly supervised learning, and reinforcement learning.

Supervised learning is the most extensively used algorithm in the field of fault diagnosis. It employs a training set comprising samples with known characteristics to construct a mathematical model. This model is then applied to predict and infer the characteristics of unknown samples. Common algorithms in this category include decision trees, artificial neural networks, support vector machines (SVMs), naive Bayes classifiers, K-nearest neighbors, logistic regression, and random forests. Antonelli et al. (2004) proposed a fault diagnosis observer for AUV actuators based on SVMs. Tun et al. (2021) combined SVM with the random forest method and proposed a hybrid fault diagnosis method called HRF-SVM. This method can be better applied to deal with small samples of fault training data. Wu and Xiong (2021) proposed a K-nearest neighbor fault monitoring method using multi-block information extraction and Mahalanobis distance, which can mine the cumulative information implied in the original data and improve the fault monitoring performance.

Unsupervised learning does not label the information of training samples. It aims to discover the inherent connections and patterns in the data through learning from unlabeled samples, which provides a basis for further data analysis. Common algorithms include clustering and dimensionality reduction algorithms. Sun et al. (2019) proposed a fault diagnosis method for AUV thrusters that integrates a deep neural network (DNN) and a denoising self-encoder. Therefore, it combines the advantages of supervised and unsupervised learning. Pan et al. (2021) proposed an online diagnostic method based on Bayesian theory for the leakage detection of deep-sea manned submersibles.

Weakly supervised learning is a method that falls between the two aforementioned methods. Its application in fault diagnosis is most notable in scenarios where obtaining labeled data is costly or achieving accurate labeling is difficult. Chang et al. (2024) proposed a weakly supervised anomaly detection network based on feature map transformation and hypersphere transformation. This method utilizes knowledge of unlabeled sample features and labeled anomaly sample features. It sets an inverse square norm loss for training and effectively distinguishes between normal and abnormal samples.

Reinforcement learning operates on the basic principle of continuous interaction between an agent and its environment. It involves obtaining state information and reward functions fed back from the environment, guiding the action of the agent, and repeatedly optimizing its action strategy through trial-and-error training to maximize rewards (Geng

et al., 2018; Chen et al., 2007; Watkins, 2004). Classical reinforcement learning algorithms, including Q-learning and Sarsa, have been established by introducing a value function to quantitatively assess the quality of the strategy. However, early reinforcement learning algorithms were limited by the computational and storage capacity of the hardware and were only suitable for low-dimensional problems. The extensive development of deep learning in recent years has enabled it to extract high-level features from high-dimensional raw data, such as images and text. This capability has also driven the development of reinforcement learning through the emergence of deep reinforcement learning (Ferguson and Pope, 2000; Yeo, 2007; Yoshida et al., 2013). Deep reinforcement learning (DRL) is a machine learning method that combines the perceptual capability of deep learning and the decision-making capability of reinforcement learning. It has great potential for application in fault diagnosis (Arulkumaran et al., 2017). Wen et al. (2022) employed a convolutional neural network for fault diagnosis and used the deep deterministic policy gradient (DDPG) algorithm for automatic optimization of the hyperparameters of the convolutional neural network. Ding et al. (2019) adopted the deep Q network (DQN) algorithm to address the fault diagnosis problem of rotating machinery. This algorithm relies on feature extraction techniques, does not require complex network structure design and parameter tuning, and can still realize the mapping of raw data to fault types.

Gathering complete fault data is difficult. Thus, conducting fault analysis and prediction based on limited fault data will be an important trend in future development (Zhai et al., 2021). We propose a fault diagnosis model based on DRL to reduce the dependence on expert experience and prior knowledge, improve the autonomy of fault diagnosis models, and achieve accurate classification and recognition of fault types through fewer fault samples. This model realizes the nonlinear mapping of raw residual data to fault modes. The highlights of the proposed method are outlined as follows: 1) A synovial observer for a complex six-degree-of-freedom X-rudder AUV is constructed to provide a training sample set for the fault diagnosis model. 2) A "Troubleshooting Quiz Game" model is designed to adapt to the operating mechanism of reinforcement learning. 3) An X-rudder fault diagnosis method based on DQN for DRL is proposed, which can achieve end-to-end fault diagnosis. 4) The DQN algorithm is improved using dropout technology, which enhances the convergence speed and accuracy of the algorithm. The rest of the paper is organized as follows. Second 2 introduces fault modeling and the sliding mode observer (SMO). Section 3 presents the principle and improvement of the proposed algorithm. Section 4 validates the effectiveness of the proposed algorithm and compares it with several other machine learning algorithms.

2 Modeling

2.1 AUV modeling

Following Wang et al. (2010), the kinematics and dynamics models are expressed as Equation (1).

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \boldsymbol{R}(\boldsymbol{\eta})\boldsymbol{v} \\ \boldsymbol{M}\dot{\boldsymbol{v}} + \boldsymbol{C}(\boldsymbol{v})\boldsymbol{v} + \boldsymbol{D}(\boldsymbol{v})\boldsymbol{v} + \boldsymbol{g}(\boldsymbol{\eta}) &= \boldsymbol{\tau} \end{aligned} \tag{1}$$

where $\boldsymbol{\eta}$ is the current position and attitude angle, $\boldsymbol{\eta} = \left[x, y, z, \varphi, \theta, \psi\right]^{\mathrm{T}}$; $\boldsymbol{R}(\boldsymbol{\eta})$ is the Jacobian matrix between $\boldsymbol{\eta}$ and $\boldsymbol{v}$; $\boldsymbol{v}$ is the velocity (linear velocity and angular velocity), $\boldsymbol{v} = \left[u, v, w, p, q, r\right]^{\mathrm{T}}$; $\boldsymbol{M}$ is the mass matrix of the AUV, $\boldsymbol{M} \in \mathbb{R}^{6\times 6}$; $\boldsymbol{C}(\boldsymbol{v})$ is the Coriolis force coefficient matrix, $\boldsymbol{C}(\boldsymbol{v}) \in \mathbb{R}^{6\times 6}$; $\boldsymbol{D}(\boldsymbol{v})$ is the viscous hydrodynamic coefficient matrix, $\boldsymbol{D}(\boldsymbol{v}) \in \mathbb{R}^{6\times 6}$; $\boldsymbol{g}(\boldsymbol{\eta})$ is the restoring force/torque vector, $\boldsymbol{g}(\boldsymbol{\eta}) \in \mathbb{R}^{6\times 1}$; $\boldsymbol{\tau}$ is the control input vector, $\boldsymbol{\tau} = \left[\tau_u, \tau_v, \tau_w, \tau_p, \tau_q, \tau_r\right]^{\mathrm{T}}$.

2.2 X-rudder and fault modeling

The rudder of an AUV generally comprises three parts: the driving unit, the transmission mechanism, and the rudder blade. After the driving unit receives steering control instructions, it rotates to generate torque. Under the action of torque, the transmission mechanism drives the rudder blade to rotate around the rudder shaft, which results in changes in the rudder.

An X-rudder AUV is a typical representative of an over-actuated control system in attitude space. The system relies on its X-shaped stern rudder to provide the torque for attitude angle control. Each rudder blade has a separate servo connected to it, which enables rudder blades to be maneuvered independently. The stern of an X-rudder is shown in Figure 1. The figure shows that the four rudders of the stern are completely consistent in material, roughness, and size. The rudder range is −25° to 25°, and the rudder rate range is $-10(^{\circ})/\mathrm{s}-10(^{\circ})/\mathrm{s}$. The number and positive direction for each rudder are given as $\boldsymbol{\delta} = \left[\delta_1, \delta_2, \delta_3, \delta_4\right]^{\mathrm{T}}$.

The actuating forces and moments generated by the

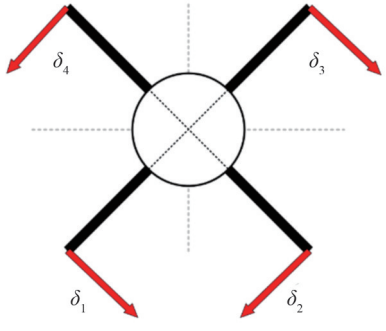

Figure 1 Layout of an X-rudder

X-rudder are denoted as Equation (2).

$$\begin{cases} \tau_u = X_{\delta_1\delta_1} u^2 {\delta_1}^2 + X_{\delta_2\delta_2} u^2 {\delta_2}^2 + X_{\delta_3\delta_3} u^2 {\delta_3}^2 + X_{\delta_4\delta_4} u^2 {\delta_4}^2 \\ \tau_v = Y_{\delta_1} u^2 \delta_1 + Y_{\delta_2} u^2 \delta_2 + Y_{\delta_3} u^2 \delta_3 + Y_{\delta_4} u^2 \delta_4 \\ \tau_w = Z_{\delta_1} u^2 \delta_1 + Z_{\delta_2} u^2 \delta_2 + Z_{\delta_3} u^2 \delta_3 + Z_{\delta_4} u^2 \delta_4 \\ \tau_p = K_{\delta_1} u^2 \delta_1 + K_{\delta_2} u^2 \delta_2 + K_{\delta_3} u^2 \delta_3 + K_{\delta_4} u^2 \delta_4 \\ \tau_q = M_{\delta_1} u^2 \delta_1 + M_{\delta_2} u^2 \delta_2 + M_{\delta_3} u^2 \delta_3 + M_{\delta_4} u^2 \delta_4 \\ \tau_r = N_{\delta_1} u^2 \delta_1 + N_{\delta_2} u^2 \delta_2 + N_{\delta_3} u^2 \delta_3 + N_{\delta_4} u^2 \delta_4 \end{cases} \tag{2}$$

where $X_{\delta_i\delta_i}, Y_{\delta_i}, Z_{\delta_i}, K_{\delta_i}, M_{\delta_i}, N_{\delta_i}$ are the nominal coefficients following Prestero (2001).

The drive unit is usually equipped with sensors (e. g., current sensors) to indicate its fault conditions. The fault forms of the transmission mechanism include fractures and stagnation, and their effects on the rudder angle are shown through the rudder angle sensor. For the rudder blade, the main forms of failure are loss and deviation (Shao et al., 2020), which are difficult to directly observe through sensors and are usually hidden. Thus, fault diagnosis methods are essential.

The desired rudder of the X-rudder mechanism is defined as $\delta_i^C (i = 1, 2, 3, 4)$, and the actual effective rudder is denoted as $\delta_i^E (i = 1, 2, 3, 4)$. The fault modeling of the X-rudder mechanism can be expressed as follows:

$$\delta_i^E = \alpha \delta_i^C + \beta \tag{3}$$

where α is used to characterize multiplicative faults, which means that the actual effective rudder is proportional to the desired rudder angle. Therefore, the following condition is satisfied: $0 \leqslant \alpha \leqslant 1$. β is used to characterize additive faults, and it indicates the offset value of the actual effective rudder relative to the desired rudder.

From Equation (3), we have the following definitions:

$\alpha = 1, \beta = 0$, which indicates no fault; $0 < \alpha < 1, \beta = 0$, which implies a lost rudder; $\alpha = 0, \beta$ (a constant value), which suggests rudder jamming; $\alpha = 1, \beta$ (a constant value), which means rudder deviation; $\alpha = 0, \beta = 0$, which indicates a complete rudder detachment or loose fault and is considered a special case of a missing rudder (i. e., a complete rudder loss).

The jammed rudder belongs to explicit faults, which can be directly detected via the rudder angle feedback from the sensor. Therefore, these faults are not within the scope of this study. This work only focuses on the diagnosis of the two hidden faults: rudder loss and rudder deviation (off-center).

2.3 Sliding mode observer

The SMO can output the AUV estimated state. The residual data are obtained by calculating the difference between the estimated state and the measured state. The training sample set can be constructed using the residual data from faulty and normal states.

The design process of the SMO is as follows. Gravity and buoyancy are assumed to cancel each other, and no external interference is considered. Under these conditions, the six-degree-of-freedom spatial equations of motion for the X-rudder AUV can be simplified to the following :

$$\boldsymbol{M}(\dot{\boldsymbol{X}})\ddot{\boldsymbol{X}} = \boldsymbol{F}_{\text{vis}}(\dot{\boldsymbol{X}}) + \boldsymbol{F}_t \tag{4}$$

where $\boldsymbol{M}$ is the mass matrix, $\ddot{\boldsymbol{X}}$ is the (angular) acceleration matrix, $\dot{\boldsymbol{X}}$ is the (angular) velocity matrix, $\boldsymbol{X}$ is the position and attitude angle matrix, $\boldsymbol{F}_{\text{vis}}$ is the viscous hydrodynamic matrix, and $\boldsymbol{F}_t$ is the control force matrix.

Let $\boldsymbol{x}_1 = \boldsymbol{X}, \boldsymbol{x}_2 = \dot{\boldsymbol{X}}$. Then, Equation (4) can be written as

$$\begin{cases} \dot{\boldsymbol{x}}_1 = \boldsymbol{x}_2 \\ \dot{\boldsymbol{x}}_2 = \boldsymbol{M}^{-1}\left[\boldsymbol{F}_{\text{vis}}(\boldsymbol{x}_2) + \boldsymbol{F}_t\right] \end{cases} \tag{5}$$

The nonlinear SMO is designed as follows:

$$\begin{cases} \dot{\hat{\boldsymbol{x}}}_1 = \hat{\boldsymbol{x}}_2 + \boldsymbol{\lambda}_1 \tanh\left(\boldsymbol{x}_1 - \hat{\boldsymbol{x}}_1\right) \\ \dot{\hat{\boldsymbol{x}}}_2 = \boldsymbol{M}^{-1}\left[\boldsymbol{F}_{\text{vis}}(\mathbf{x}_2) + \boldsymbol{F}_t\right] + \boldsymbol{\lambda}_2 \tanh\left(\boldsymbol{x}_1 - \hat{\boldsymbol{x}}_1\right) \end{cases} \tag{6}$$

where $\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2$ is an estimate of $\boldsymbol{x}_1, \boldsymbol{x}_2$. $\boldsymbol{\lambda}_1 = \text{diag}\left(\lambda_{11}, \lambda_{12}, \cdots, \lambda_{16}\right)$, $\boldsymbol{\lambda}_2 = \text{diag}\left(\lambda_{21}, \lambda_{22}, \cdots, \lambda_{26}\right)$, and $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2 > 0$. The hyperbolic tangent function is used instead of the sign function in the classical SMO because the abrupt type switching of the sign function near the zero point leads to a jitter in the observer output near the sliding surface.

Observer errors are defined as

$$\begin{cases} \boldsymbol{e}_1 = \boldsymbol{x}_1 - \hat{\boldsymbol{x}}_1 \\ \boldsymbol{e}_2 = \boldsymbol{x}_2 - \hat{\boldsymbol{x}}_2 \end{cases} \tag{7}$$

According to Equations (5) (6) and (7), the error output of the state observer is shown in Equation (8).

$$\begin{cases} \dot{\boldsymbol{e}}_1 = \boldsymbol{e}_2 - \boldsymbol{\lambda}_1 \tanh(\boldsymbol{e}_1) \\ \dot{\boldsymbol{e}}_2 = -\boldsymbol{\lambda}_2 \tanh(\boldsymbol{e}_1) \end{cases} \tag{8}$$

According to Wang et al. (2005), choosing a proper gain matrix $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2$ results in $\boldsymbol{e}_1^{\mathrm{T}} \dot{\boldsymbol{e}}_1 < 0$. The SMO satisfies the attractiveness condition, which allows the sliding residuals to converge to the sliding surface. Thus, $\boldsymbol{e}_1 \rightarrow 0, \dot{\boldsymbol{e}}_1 \rightarrow 0$.

When the actuator fails, the deviation between the actual output of the system and the output of the observer increases gradually. The corresponding sliding residual will deviate from the zero point, the oscillatory characteristic then disappears, and the sliding mode stops immediately. Accordingly, the residual data can be applied to fault diagnosis.

3 Design and optimization of algorithms

Fault diagnosis, as a pattern recognition problem, is typically addressed using machine learning methods. Supervised learning-based fault diagnosis methods are the most widely applied. These methods require the simultaneous input of fault data and corresponding labels to distinguish between different fault modes. For reinforcement learning-based methods, the performance of fault diagnosis agents is assessed by overall rewards. This weaker feedback provides much less guidance for performance adjustments compared with the feedback used in supervised learning methods. In reinforcement learning, agents usually only receive overall reward signals without feedback for each sample. Consequently, agents may not know which specific samples lead to overall poor performance. They must indirectly infer effective strategies or behaviors and identify areas for improvement through overall reward signals. This learning approach requires agents to explore and identify patterns and differences in the data on their own without clear instructions, which improves their overall performance.

3.1 Troubleshooting quiz game

Reinforcement learning theory enables agents to interact with their environment and choose actions based on a strategy to maximize gains. The agent accumulates extensive learning experience in the environment, learns from its mistakes, and updates its strategy (Xing et al., 2024). The interaction process between the agent and the environment is usually represented by a Markov decision process (MDP).

At any time step t, the agent is in the current environment state s_t. When the agent selects an action in the environment, a reward value r_t will be provided. The state of the environment is updated to the next state s_{t+1} according to the state transfer function $p_{ss'}^{a}$. The reinforcement learning algorithm selects the action for the agent by means of an action selection policy $\pi(a|s)$. Therefore, the MDP can be expressed as $\langle S, A, P, R, \gamma \rangle$, where γ is an attenuation factor, $\gamma \in [0, 1]$ (Fang et al., 2012; Wang and Fang, 2023).

According to MDP, an environment–agent interaction model is designed to transform supervised learning problems into reinforcement learning problems. The "Troubleshooting Quiz Game" is set as follows:

1) Select a random strip of sample data (X_t, Y_t) from the troubleshooting sample set.

2) Pass sample feature X_t as the state of the environment at the current time step s_t to the agent (asking the question).

3) The agent performs the action a_t to state s_t (answering the question).

4) The environment receives the action a_t and compares it with the true fault type of the sample Y_t and provides a reward r_t.

5) Proceed to the next time step, take the next piece of sample data from the troubleshooting sample set at random (X_{t+1}, Y_{t+1}), and continue running the Q&A session.

6) A round ends upon the completion of T time steps of environment–agent interaction. The cumulative reward R of the episode is regarded as an evaluation metric for the correctness of the agent's troubleshooting in that episode.

The procedure is shown in Figure 2.

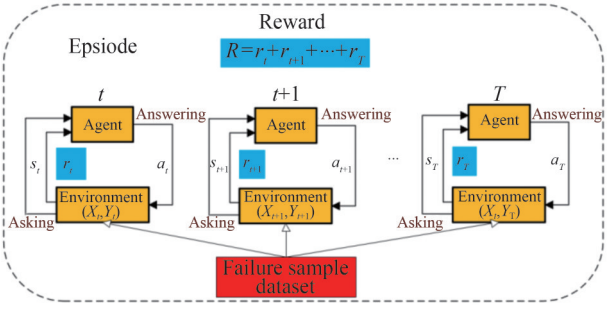

Figure 2 Procedure of "Troubleshooting Quiz Game" in one episode

where $\boldsymbol{X}_t$ is the state space, $\boldsymbol{X}_t = [u_e, v_e, w_e, p_e, q_e, r_e, \varphi_e, \theta_e, \psi_e]$. a_t is the action space, $Y_t, a_t \in A, A = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$. The reward function is defined as follows: $r_t = \begin{cases} 1 & a_t = Y_t \\ 0 & a_t \neq Y_t \end{cases}$.

3.2 Q-learning

Q-learning is a classic model-free reinforcement learning algorithm. It is also considered a type of value function learning method, where "Q" stands for "quality". The Q-function signifies the quality of executing an action in a given state, and it is defined as

$$Q_\pi(s, a) = E\left[r' + \gamma r'' + \gamma^2 r''' + \cdots | s, a\right] \tag{9}$$

where $Q_\pi(s, a)$ is the Q value for state s given action a, r is the reward of each time step, and γ is the discount rate.

The Q-learning algorithm creates a table to calculate the maximum expected future reward for each state and action; this table is known as the Q-table. The Q-table is the data structure used in the Q-learning algorithm to store Q-values, and it is a two-dimensional array. The rows correspond to all possible states, and the columns correspond to all possible actions. Each element $Q(s, a)$ represents the expected return of taking action a in state s.

The Q-learning algorithm is outlined as follows:

Initialize: Start with a Q-table full of zeros, where each cell represents the value of taking a specific action in a specific state.

Choose action: For each state, determine an action based on the current Q-table while balancing exploration (trying random actions) and exploitation (choosing the best-known action).

Perform and learn: Take action and observe the reward and the new state. Update the Q-value for the taken action using the following formula:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right] \tag{10}$$

where α is the learning rate, r is the reward for taking action a at state s, and $\max_{a'} Q(s',a')$ is the maximum expected future reward given the new state s' and all possible actions.

Repeat: Repeat the process for multiple episodes until the Q-values stabilize, which indicates that the agent has learned a good policy.

Policy extraction: After learning, the agent can follow the policy by selecting actions with the highest Q-values for each state.

In general, Q-learning involves learning the state–action function according to the agent's interactions with the environment, continuously updating the Q-table, and ultimately obtaining the optimal policy based on the converged Q-table.

3.3 DQN algorithm

However, Q-learning becomes infeasible when dealing with large state spaces, given that the size of the Q-table grows exponentially with the number of states and actions. In such cases, the computational cost of the algorithm is high, and a large amount of memory is required to store the Q-values.

Mnih et al. (2013) combined Q-learning with DNNs to address the aforementioned challenge. This method is known as DQN, where the neural network in DQN acts as an approximator for the Q-values of each (state, action) pair. DQN is a newly developed end-to-end reinforcement learning method that uses a DNN to map the relationships between actions and states, which is similar to the Q-table in Q-learning. The neural network takes states as inputs and outputs the Q-values for all possible actions. Figure 3 illustrates the difference between Q-learning and DQN in evaluating Q-values:

Essentially, DQN replaces the conventional Q-table with a neural network. Instead of mapping (state, action) pairs to Q-values, the neural network maps input states to (action, Q-value) pairs. In the training process of the DQN, two modifications of Q-learning are made to ensure that the training process of DQN does not diverge: experience replay and establishment of target network.

3.3.1 Experience replay

Experience replay is an efficient technique for preventing oscillations or divergence in the parameters. It smoothens the learning process by allowing the agent to consider its experience. The experience replay mainly consists of

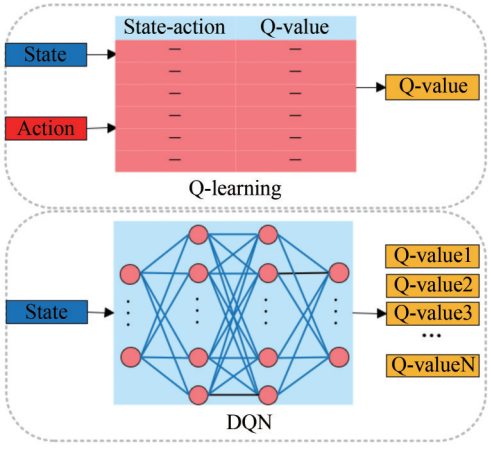

Figure 3 Difference between Q-learning and DQN

two key steps: "storage" and "sampling and replay".

Storage: During the interaction with the environment, the agent collects a series of experience tuples, each containing the current state s, the action taken a, the reward received r, and the next state s'. These experiences (s, a, r, s') are stored in a data structure called the "experience replay buffer".

Sampling and replay: During the training process, the agent randomly draws a batch of experiences from the experience replay buffer and uses these experiences to train the neural network.

Experience replay can eliminate data correlation, which makes the data more likely independent and identically distributed. This reduction in data can decrease the variance in parameter updates and accelerate convergence.

3.3.2 Target network

The DQN algorithm employs iterative updates. These updates adjust the action values to target values that are only periodically updated. This process reduces the correlation with the target. The introduction of a target network into DQN algorithms is accomplished by building two neural networks with the same structure: the target network and the evaluation (eval) network (Mnih et al., 2015).

The eval network uses the current state of the environment to compute the Q-values for all possible actions within the action space. The target network is used to calculate the target-Q value with its output $Q_{\theta'}(s',a')$, and then update the eval network with the target-Q value:

$$y = r + \gamma \max_{a'} Q\left(s', a'; \theta^{-}\right) \tag{11}$$

where y is the target-Q value, and θ^{-} is the parameter of the target network.

The algorithm utilizes the difference between the two datasets to calculate the loss function of the neural network. The loss function is used to update the parameters of the neural network for Q-eval, which determines the

speed of convergence of the neural network. The loss function for the parameter of eval network update is

$$L(\theta) = E\left[\left(y - Q(s, a\,;\,\theta)\right)^2\right] \tag{12}$$

where θ is the parameter of the eval network.

During the training of the algorithm, the parameters of the eval network are updated by the gradient descent method based on Equation (12). Then, two neural networks perform parameter synchronization at intervals through $\theta \rightarrow \theta^-$.

3.3.3 Greedy policy

The DQN algorithm balances the exploration and utilization of reinforcement learning through a $\varepsilon-greedy$ strategy based on the principle of

$$a = \begin{cases} \forall a \in A & p \in [0, \varepsilon] \\ \underset{a \in A}{\operatorname{argmax}}\, Q_\theta(s, a) & p \in [\varepsilon, 1] \end{cases} \tag{13}$$

where p is a random number, and ε is the parameter for balancing exploration and utilization. Clearly, the value of ε significantly impacts the training efficiency of the algorithm. The traditional approach is to set it to a small value near 0, which results in low exploration and high exploitation. However, this approach can cause slower convergence in practice. Therefore, the following mechanism for the dynamic adjustment of ε is designed:

$$\varepsilon = \max\left(0.99^{n_{\text{episode}}}\varepsilon_0, \varepsilon_{\min}\right) \tag{14}$$

where ε_0 is the initial value, n_{episode} is the number of training rounds, and $\varepsilon_{\min}$ is the minimum value. This approach allows for a higher probability of exploration during the initial training phase, which accelerates the convergence of the algorithm. During the later stages of training, it reduces the probability of exploration to prevent excessive exploration.

The pseudo-code of the DQN is shown in Algorithm 1.

Algorithm 1 DQN with Experience Replay

Initialize e*val network, target network* with random weights θ, θ^-
Initialize replay memory B
for n_{episode}=1 **to** M **do**
  get features s_0 of start state s of current episode
  Get features ε of current episode with $\varepsilon = \max\left(0.99^{n_{\text{episode}}}\varepsilon_0, \varepsilon_{\min}\right)$
  for t=1 **to** T **do**
    Select a with $\varepsilon-greedy$ policy
    Execute action a and observe reward r and s
    Store transition (s, a, r, s') in B
    Sample random minibatch of transition (s, a, r, s')
    Set $y = r + \gamma \max_{a'} Q\left(s', a';\, \theta^-\right)$
    Perform a gradient descent on $L(\theta) = E\left[\left(y - Q(s, a\,;\,\theta)\right)^2\right]$ to update θ
    If t **mod** C **then** // timing synchronization parameter
      set $\theta^- = \theta$
    end if
  end for
end for

3.4 DQN-D algorithm

Given that faults are rare states during system operation, the difficulty for the data-driven fault diagnosis method for AUVs lies in the small number of AUV fault samples and the imbalance between the number of faults and normal samples. This practical situation can easily lead to network training overfitting. In this subsection, the dropout technique is integrated into the DQN algorithm to prevent the overfitting phenomenon from affecting the accuracy of fault diagnosis. The DQN algorithm, which is improved with the dropout technique, is referred to as the Deep Q-network-dropout (DQN-D) algorithm (Hinton et al., 2012).

The DQN algorithm has a fully connected neural network with two hidden layers, each containing 128 neurons. The hidden layer adopts the Relu activation function. The input to the neural network is a 9-dimensional state vector, and the output is the Q-value corresponding to 9 discrete actions. Figure 4 shows the DQN neural network structure applying the dropout technique. During the training phase, dropout is applied to the hidden layers to randomly eliminate the activation values of some neurons during forward propagation. For each neuron in every hidden layer, a random vector of the same size as the neuron's output is generated based on the dropout probability. It determines whether to set its output to zero. During back propagation (BP), only the neurons that were not dropped out during the forward propagation participate in the gradient computation and weight updates.

The principle of the dropout technique is given in the Equation (15):

$$\begin{aligned} r_j^l &\sim \text{Bernoulli}(p) \\ \tilde{\boldsymbol{y}}^l &= \boldsymbol{r}^l \boldsymbol{y}^l \\ z_i^{l+1} &= w_i^{l+1} \tilde{\boldsymbol{y}}^l + b_i^{l+1} \\ y_i^{l+1} &= f\left(z_i^{l+1}\right) \end{aligned} \tag{15}$$

where p is the probability that the output of a neuron is randomly zeroed; $\boldsymbol{r}^l$ is the vector of $0-1$ according to Bernoulli; $\boldsymbol{y}^l$ is the output of the l layer of the neural network; $\tilde{\boldsymbol{y}}^l$ is the result of randomly zeroing $\boldsymbol{y}^l$; w_i^{l+1} and

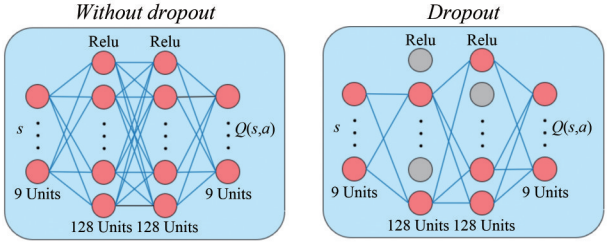

Figure 4 Structure of the neural network in DQN-D

$b_i^{l+1}$are the weight and bias of the i neuron in the $l+1$ layer of the neural network; $f(\cdot)$ is the activation function; y_i^{l+1} is the output of the neuron in the $l+1$ layer of the neural network.

The structure of the proposed DQN-D is shown in Figure 5.

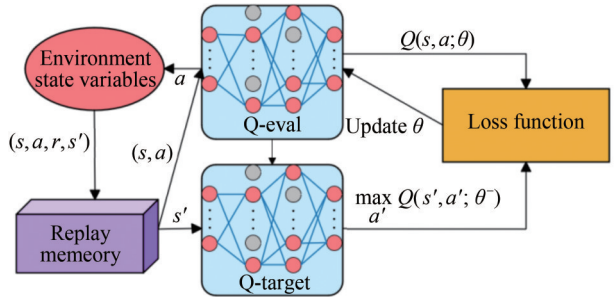

Figure 5 Structure of the DQN-D algorithm

4 Simulations

4.1 Training sample

According to the SMO in Subsection 2.3, the sample set is obtained by simulating the fault state with rudder loss and rudder deviation, setting the control task as variable depth and variable bow control, and calculating the difference between the actual outputs during the motion control process and the estimated outputs from the SMO.

The residual signal is $\boldsymbol{X}_t = \left[u_e, v_e, w_e, p_e, q_e, r_e, \varphi_e, \theta_e, \psi_e\right]$. The troubleshooting sample set for the X-rudder AUV rudder mechanism is described in Table 1.

The control objectives remain consistent for normal and fault states, with the single rudder blade loss set to 50% and the single rudder blade deviation fault set to 5°. The simulation results of residual samples for δ_1 are shown in Figure 6.

4.2 Analysis of simulation results

To verify the effectiveness of the DQN-D algorithm designed in this study, the fault diagnosis method based on the DQN-D algorithm will be used to train and test the fault diagnosis model for the X-rudder AUV rudder mechanism. The sample set is divided into a training set and a test set in a 9∶1 ratio for fault diagnosis model training. Two sets of comparison tests are conducted: 1) the performance of the DQN-D fault diagnosis model with different dropout rates is compared to validate the effect of the dropout technique on the DQN algorithm; 2) the performance of the DQN-D algorithm, BP neural network, and SVM is compared to assess the relative advantages and disadvantages of the DQN-D algorithm over other machine learning fault diagnosis methods. The abovementioned algorithms use the same dataset, training, and testing process in the comparison.

Table 1 Troubleshooting sample set for the rudder mechanism

Fault type	Fault description	Sample size	Label
F0	Normality	2 000	0
F1	Rudder 1 loss	1 000	1
F2	Rudder 1 deviation	1 000	2
F3	Rudder 2 loss	1 000	3
F4	Rudder 2 deviation	1 000	4
F5	Rudder 3 loss	1 000	5
F6	Rudder 3 deviation	1 000	6
F7	Rudder 4 loss	1 000	7
F8	Rudder 4 deviation	1 000	8

4.2.1 Performance comparison of DQN-D fault diagnosis models with different dropout rates

The dropout rate (p) is an important hyperparameter of the DQN-D algorithm. It represents different probabilities of neuron failure during the model training. Different dropout rates affect the speed and performance of model convergence, therefore. Thus, six fault diagnosis models are simulated and trained with different dropout rates to compare their fault diagnosis accuracy. In addition to the dropout rate, the hyperparameters of the DQN-D algorithm are shown in Table 2.

In the "Troubleshooting Quiz Game", only the correctness of the current sample judgment is valued. The learning rate controls the speed of weight updates in neural networks. Considering the limited sample size, a lower learning rate of 0.003 is chosen to prevent overfitting during

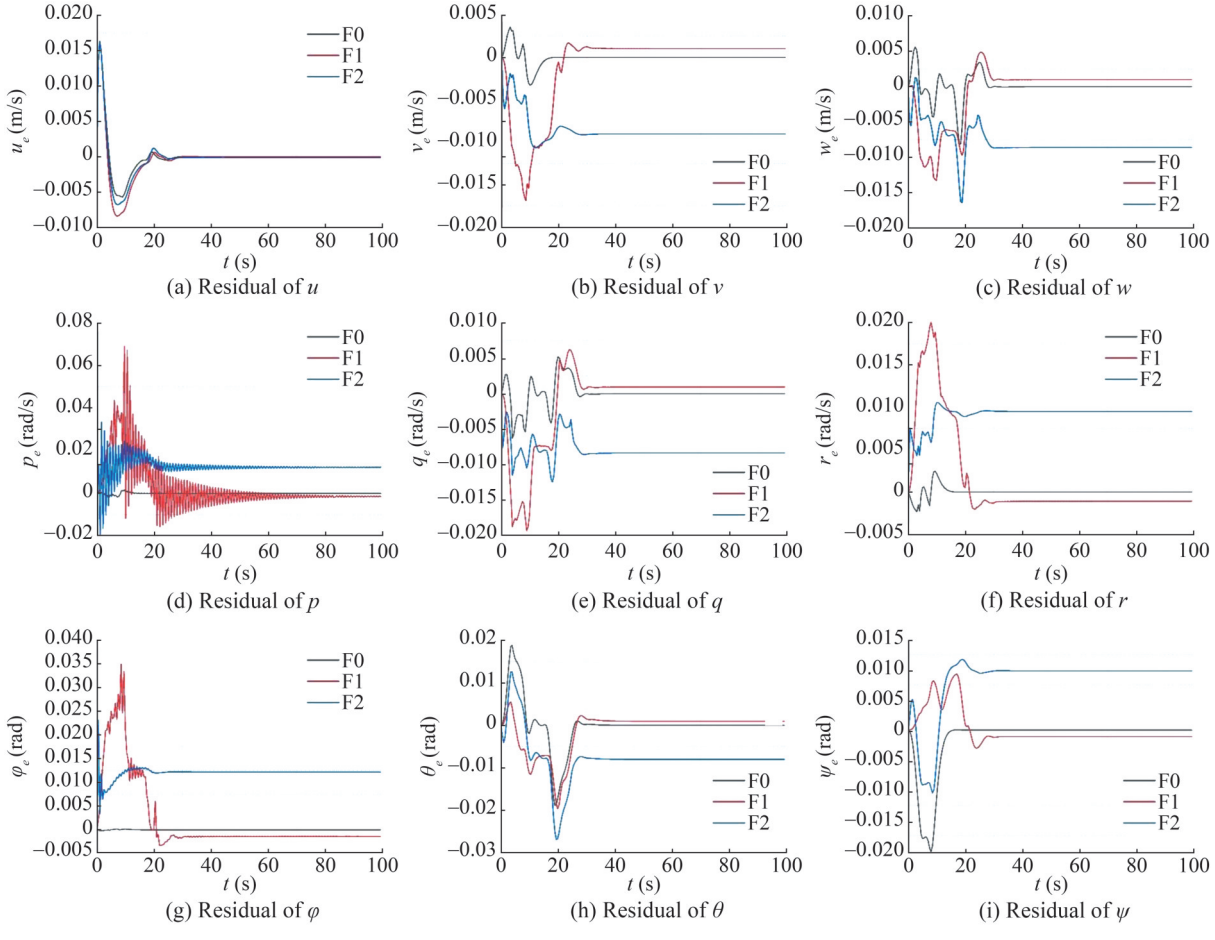

Figure 6 Residual data simulation results of δ_1

Table 2 Algorithm training parameters

Parameters	Value
Discount factors γ	0
Eval network learning rate	0.000 3
Initial greedy value ε_0	1
Min greedy value $\varepsilon_{\min}$	0.1
Target network update frequency C	10
Batch size N	128
Replay capacity B	10 000
Max episode T	100

training. Initial greedy value ε_0 and minimum greedy value $\varepsilon_{\min}$ determine the balance between exploration and exploitation for the agent. Notably, higher values encourage exploration to accelerate convergence in the early stages, and smaller values prevent excessive exploration in the later stages. The target network update frequency C, batch size N, replay capacity B, and max episode T are determined by the training cost of the model, the complexity of the task, and the depth learned. In this study, the size of these hyperparameters is obtained through multiple experiments to determine the optimal value.

The sliding average cumulative reward curve is shown in Figure 7.

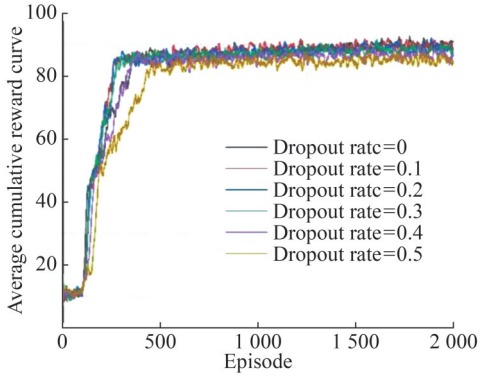

Figure 7 Average cumulative reward for different dropout rate models

Figure 7 shows that setting different dropout rates affects the convergence speed of the algorithm. When the dropout rate ranges from 0.1 to 0.3, the convergence speed of the

algorithm is more excellent than that of the case with a dropout rate of 0. When the dropout rate is from 0.4 to 0.5, the convergence speed of the algorithm is smaller than that of the case with a dropout rate of 0. The fastest convergence is achieved when the dropout rate is 0.2, which is approximately 84 rounds ahead of the convergence of the case with a dropout rate of 0.

Figure 8 displays the confusion matrices for different dropout rates, where the rows in the confusion matrix represent the true labels, and the columns represent the labels predicted by the model. The nine rows and nine columns correspond to nine states of residual data. The experiment is conducted on a test set of 1 000 samples. The graph shows a general tendency to misdiagnose various fault types as normal labels. However, most models perform well, with the fewest errors when the dropout rate is 0.2 and the most errors when the dropout rate is 0.5.

To further analyze the performance of the fault diagnosis models with different dropout rates, the precision, recall, and $F_{1\text{-score}}$ of the model results are calculated as follows:

$$\begin{aligned} A_{\text{ccuracy}} &= \frac{T_{\text{P}} + T_{\text{N}}}{T_{\text{P}} + T_{\text{N}} + F_{\text{P}} + F_{\text{N}}} \\ P_{\text{recision}} &= \frac{T_{\text{P}}}{T_{\text{P}} + F_{\text{P}}} \\ R_{\text{ecall}} &= \frac{T_{\text{P}}}{T_{\text{P}} + F_{\text{N}}} \\ F_{1\text{-score}} &= \frac{2 \times P_{\text{recision}} \times R_{\text{ecall}}}{P_{\text{recision}} + R_{\text{ecall}}} \end{aligned} \tag{16}$$

where T_{P}, T_{N}, F_{P}, and F_{N} represent the number of true positive, true negative, false positive, and false negative predictions, respectively. Accuracy represents the overall predictive power of the model and is the most intuitive performance metric. Precision represents the ability to avoid false positive predictions, recall represents the ability to correctly identify positive instances, and $F_{1\text{-score}}$ represents the overall balance between precision and recall. Table 3 describes the details of precision rates, recall rates, $F_{1\text{-score}}$ rates, and accuracy for the experimental results according to Equation (16) (Xu et al., 2024).

As shown in Table 3, even at a dropout rate of 0, all four performance metrics are above 94%. When the dropout rate is between 0.1 and 0.4, the four performance metrics of the models are all greater than those at a dropout rate of 0. At a dropout rate of 0.2, the four performance metrics reach their maximum values with the highest average accuracy. As the dropout rate continues to increase, the performance metrics of the models gradually decrease. When the dropout rate is 0.5, the metrics of the models are already lower than those at a dropout rate of 0.

The abovementioned results demonstrate that introducing dropout technology into the DQN algorithm and setting an appropriate dropout rate can improve not only the convergence speed of the algorithm but also its performance. Consequently, the trained fault diagnosis model has a higher fault diagnosis accuracy. The introduction of dropout technology can effectively inhibit the issue of neural networks' tendency to overfit with a small number of fault samples. It effectively enhances the performance of the fault diagnosis model.

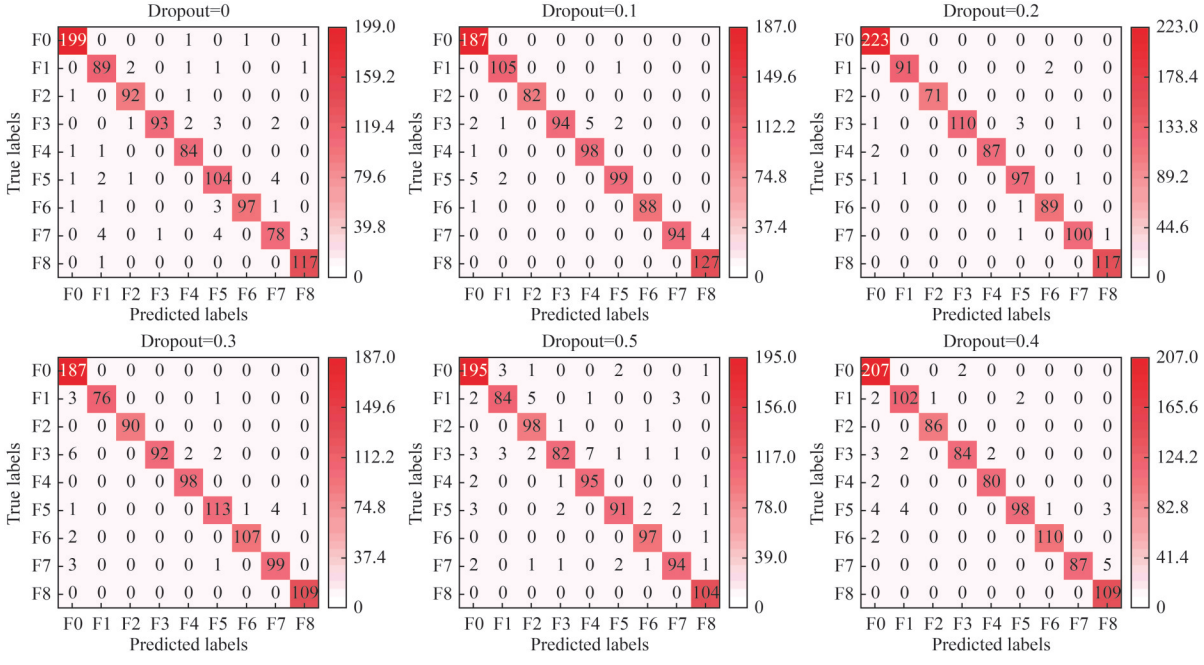

Figure 8 Confusion matrix of fault diagnosis models with different dropout rates

Table 3 Performance of models with different dropout rates

Dropout	Precision (%)	Recall (%)	$F_{1\text{-score}}$ (%)	Accuracy (%)
0	99.60	94.13	96.79	95.30
0.1	100	96.80	98.38	97.40
0.2	100	98.07	99.02	98.50
0.3	100	96.43	98.18	97.10
0.4	99.73	95.33	97.48	96.30
0.5	99.07	92.54	95.69	94.00

4.2.2 Performance comparison of fault diagnosis models based on DQN-D algorithm, BP neural networks, and SVMs

The DQN-D algorithm is compared horizontally with commonly used machine learning algorithms. The DQN-D algorithm is tested with the dropout rate of 0.2. The comparison methods are as follows: 1) BP neural network: BP-128, BP-256, and BP-512. The number of network layers is the same as that of the Q-network in the DQN algorithm, with Relu as the hidden layer activation function and softmax as the output layer activation function. The numbers of neurons in the hidden layer are 128, 256, and 512. 2) SVM: SVM-linear, SVM-poly, and SVM-rbf. The kernel functions are linear, polynomial, and radial basis functions. The comparison of different algorithmic fault diagnosis models is shown in Figure 9.

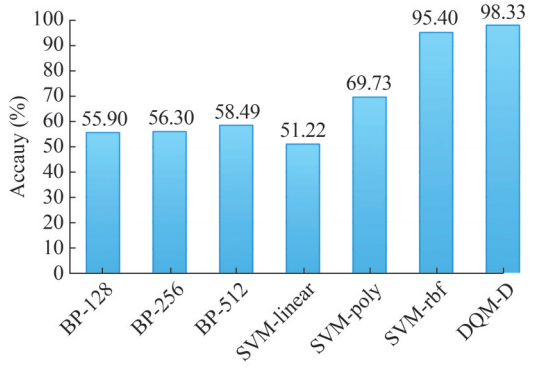

Figure 9 Comparison of different algorithmic fault diagnosis models

Figure 9 demonstrates that the DQN-D algorithm with a dropout rate of 0.2 has the highest diagnostic accuracy. The diagnostic accuracy of the model trained by the shallow BP neural network is much lower than those of the two other algorithms, which ranges only between 55% and 59%. Although increasing the number of neurons in the hidden layer improves the diagnostic accuracy, reaching an acceptable accuracy is still difficult. The SVM-based fault diagnosis model is more dependent on the kernel function. Specifically, when the kernel function is rbf, the diagnostic accuracy is higher, which is similar to that of the DQN algorithm without dropout. Thus, the two algorithms are compared separately, the comparison results are shown in Figure 10.

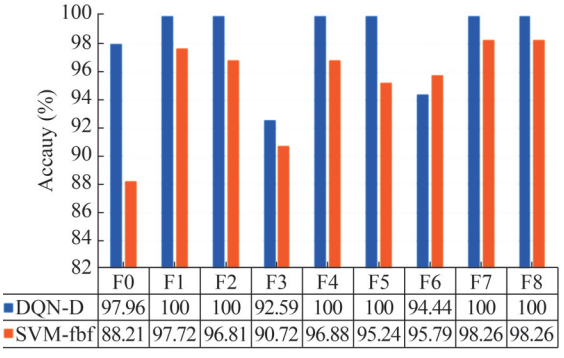

Figure 10 Comparison of SVM-rbf- and DQN-D-based fault diagnosis models

Figure 10 demonstrates that, for the fault type "F6", the diagnostic accuracy of the DQN-D-based model is 4.54% lower than that of the SVM-rbf-based model. For all other fault types, the diagnostic accuracy of the DQN-D-based model is higher than that of the SVM-rbf-based model. Even for the normal state, the diagnostic accuracy of the DQN-D-based model is 9.75% higher than that of the SVM-rbf-based model. Therefore, the fault diagnosis method based on DQN-D has a stronger fault diagnosis ability than the method based on SVM-rbf.

Based on the comprehensive analysis of the abovementioned comparative test results, the shallow BP neural network model and SVM algorithm with linear and polynomial kernel functions are unsuitable for the fault diagnosis of the X-rudder AUV steering mechanism. The main reason is that the sample size is small, and the neural network structure is too simple to extract features from limited data. The SVM with RBF is suitable for diagnosing faults in the X-rudder AUV steering mechanism, but its effectiveness is inferior to that of DQN-D, with a high misdiagnosis rate. This finding proves the advantages of reinforcement learning-based fault diagnosis methods.

5 Conclusions

5.1 Main reults

This study proposes an improved DQN algorithm-based fault diagnosis model for X-rudder AUVs based on DRL. This model addresses the drawbacks of traditional fault diagnosis methods and tackles the issue of scarce fault sample data through a series of design and optimization methods.

1) An SMO is established through the six-degree-of-freedom motion model of the X-rudder AUV, which provides a foundation for training and testing models by generating residual datasets for normal and faulty conditions.

2) A "Troubleshooting Quiz Game" is designed to transform the supervised learning problem into a reinforcement

learning problem. Through this mechanism, the model can autonomously explore the mapping relationship between raw residual data and fault patterns, which achieves end-to-end mapping.

3) The introduction of dropout technology during model training effectively overcomes the overfitting problem caused by the scarcity of data samples. This improvement leads to rapid algorithm convergence and performance enhancement.

5.2 Main limitation of the method

First, the modeling of X-rudder mechanism failure ignored the situation of multiple rudder blade failures. Then, the accuracy of the fault diagnosis model needs to be further improved. Finally, given the current stage of our research, we were unable to conduct actual experiments or test the actual performance of the proposed algorithm.

5.3 Future prospect

1) A more comprehensive fault model for the X-rudder can be established to achieve high-precision classification and recognition of all fault types.

2) Based on the existing DQN-D algorithm, performance should be further improved and training costs reduced to enable models to achieve or approach 100% accuracy.

3) Navigation data from real operations of the X-rudder AUV should be collected to verify the feasibility of the proposed method in this study.

Funding Supported by the National Natural Science Foundation of China under Grant Nos. 52071099, 52071104; National Key Project of Research and Development Program under Grant No. 2021YFC2801300; Research Fund from National Key Laboratory of Autonomous Marine Vehicle Technology under Grant No. 2023-SXJQR-SYSJJ01.

Competing interest The authors have no competing interests to declare that are relevant to the content of this article.

References

Alessandri A (2003) Fault diagnosis for nonlinear systems using a bank of neural estimators. Computers in Industry 52(3): 271-289. https://doi.org/10.1016/S0166-3615(03)00131-3

Alex Gong CS, Simon Su CH, Tseng KH (2020) Implementation of Machine Learning for Fault Classification on Vehicle Power Transmission System. IEEE Sensors Journal 20(24): 15163-15176. https://doi.org/10.1109/JSEN.2020.3010291

Antonelli G, Caccavale F, Sansone C, Villani L (2004) Fault diagnosis for AUVs using support vector machines. IEEE International Conference on Robotics and Automation 4486-4491. DOI: 10.1109/ROBOT.2004.1302424

Arulkumaran K, Deisenroth MP, Brundage M, Bharath AA (2017) Deep Reinforcement Learning: A Brief Survey. IEEE Signal Processing Magazine 34(6): 26-38. https://doi. org/10.1109/MSP.2017.2743240

Chang ZH, Jia KW, Han T, Wei YM (2024) Towards more reliable photovoltaic energy conversion systems: A weakly-supervised learning perspective on anomaly detection. Energy Conversion and Management 316: 118845. https://www. sciencedirect. com/science/article/pii/S0196890424007866

Chen Y, Mabu S, Hirasawa K, & Hu J (2007) Enhancement of trading rules on stock markets using genetic network programming with Sarsa learning. SICE Annual Conference 2007: 2700-2707. https://doi.org/10.1109/SICE.2007. 4421448

Ding Y, Ma L, Ma J, Suo M, Tao L, Cheng Y, Lu C (2019) Intelligent fault diagnosis for rotating machinery using deep Q-network based health state classification: A deep reinforcement learning approach. Advanced Engineering Informatics 42: 100977. https://doi.org/10.1016/j.aei.2019.100977

Fang M, Li H, Zhang X (2012) A Heuristic Reinforcement Learning Based on State Backtracking Method. 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, 673-678. https://doi.org/10.1109/WI-IAT.2012.187

Ferguson J, Pope A (2000) Explorer-a modular AUV for commercial site survey. Proceedings of the 2000 International Symposium on Underwater Technology (Cat. No. 00EX418). IEEE, 129-132

Frank PM (1990) Fault Diagnosis in Dynamic Systems Using Analytical and Knowledge-based Redundancy A Survey and Some New Results. Automatica 26(3): 459-474 https://doi.org/10.1016/0005-1098(90)90018-D

Geng H, Liu H, Wang B, Sun F (2018) Reinforcement Extreme Learning Machine for Mobile Robot Navigation. In J. Cao, E. Cambria, A. Lendasse, Y. Miche, & C. M. Vong (Eds.), Proceedings of ELM-2016, Vol. 9, pp. 61-73. Springer International Publishing, https://doi.org/10.1007/978-3-319-57421-9_6

Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR (2012) Improving neural networks by preventing co-adaptation of feature detectors (arXiv: 1207.0580) arXiv. http://arxiv.org/abs/1207.0580

Huang Y, Li Y, Yu JC, Li S, Feng XS (2020) State-of-the-Art and Development Trends of AUV Intelligence. Robot 42(02): 215-231. (in Chinese) https://doi.org/10.13973/j.cnki.robot.190392

Ji D, Yao X, Li S, Tang Y, Tian Y (2021) Model-free fault diagnosis for autonomous underwater vehicles using sequence Convolutional Neural Network. Ocean Engineering 232: 108874. https://doi.org/10.1016/j.oceaneng.2021.108874

Jiang Y, Yin S (2019) Recent Advances in Key-Performance-Indicator Oriented Prognosis and Diagnosis With a MATLAB Toolbox: DB-KIT. IEEE Transactions on Industrial Informatics 15(5): 2849-2858. https://doi.org/10.1109/TII.2018.2875067

Jiang Y, Yin S, Kaynak O (2018) Data-driven monitoring and Safety Control of Industrial Cyber-Physical Systems: Basics and Beyond. IEEE Access 6: 47374-47384. https://doi.org/10.1109/ACCESS.2018.2866403

Liu F, Xu D (2016) Fault Localization and Fault-Tolerant Control for rudders of AUVs. 2016 35th Chinese Control Conference (CCC), 6537-6541

Miskovic N, Barisic M (2005) Fault Detection and Localization on Underwater Vehicle Propulsion Systems Using Principal Component Analysis. Proceedings of the IEEE International Symposium on Industrial Electronics, 1721-1728. https://doi.org/10.1109/ISIE.2005.1529192.

Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M, Rusu AA, Veness J, Bellemare MG, Krizhevsky A, Hinton G, Hassabis D (2013) Playing atari with deep

reinforcement learning. arXiv preprint arXiv: 1312.5602

Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D (2015) Human-level control through deep reinforcement learning. Nature 518(7540): 529-533. https://doi.org/10.1038/nature14236

Pan Y, Zheng Z, Fu D (2021) Bayesian-based water leakage detection with a novel multisensor fusion method in a deep manned submersible. Applied Ocean Research 106: 102459. https://doi.org/10.1016/j.apor.2020.102459

Prestero T (2001) Verification of a six-degree of freedom simulation model for the REMUS autonomous underwater vehicle Massachusetts Institute of Technology and Woods Hole Oceanographic Institution. https://doi.org/10.1575/1912/3040

Shao Z, Hua YX, Shi JY (2020) Research on Fault Diagnosis Method for Rudder Surface Based on Multiple Models. Flight Dynamics 38(3): 24-27. (in Chinese) https://doi.org/10.13645/j.cnki.f.d20200312.010

Sun Y, Ran X, Li Y, Zhang G, Zhang Y (2016) Thruster fault diagnosis method based on Gaussian particle filter for autonomous underwater vehicles. International Journal of Naval Architecture and Ocean Engineering 8(3): 243-251. https://doi.org/10.1016/j.ijnaoe.2016.03.003

Sun Y, Wang Z, Zhang G (2019) Fault Diagnosis Method of Autonomous Underwater Vehicle Based on Deep Learning. IOP Conference Series: Materials Science and Engineering 470: 012035

Tun W, Wong JK W, Ling SH (2021) Hybrid Random Forest and Support Vector Machine Modeling for HVAC Fault Detection and Diagnosis. Sensors 21(24): 8163. https://doi.org/10.3390/s21248163

Wang F, Wan L, Su Y, Xu Y (2010) AUV modeling and motion control strategy design. Journal of Marine Science and Application 9(4): 379-385. https://doi.org/10.1007/s11804-010-1023-4

Wang LR, Gan Y, Xu YR, Wan L (2005) Sliding-mode observers used in thruster fault diagnosis of an autonomous underwater vehicle. Journal of Harbin Engineering University 26(4): 425-429. https://doi.org/10.3969/j.issn.1006-7043.2005. 04.002

Wang X, Fang X (2023) A multi-agent reinforcement learning algorithm with the action preference selection strategy for massive target cooperative search mission planning. Expert Systems with Applications 231: 120643. https://doi.org/10.1016/j.eswa.2023.120643

Wang YJ, Zhang MJ, Che ZZ (2011) Qualitative diagnostic model for underwater robot propeller fault. Proceedings of the Seventh National Conference on Fault Diagnosis and Safety of Technical Processes

Wang YJ, Zhang MJ, Wu J (2007) Research of the fault diagnosis method for the thruster of AUV based on information fusion. Third International Conference on Agent Computing, ICIC

Watkins, CJ, Dayan, P (1992) Q-learning. Machine learning, 8, 279-292

Wen L, Wang Y, Li X (2022) A new automatic convolutional neural network based on deep reinforcement learning for fault diagnosis. Frontiers of Mechanical Engineering 17(2): 17. https://doi.org/10.1007/s11465-022-0673-7

Wu X, Xiong W (2021) kNN Fault Detection Based on Multi-block Information Extraction and Mahalanobis Distance. Information and Control 50(3): 287-296. https://doi.org/10.13976/j.cnki.xk.2021.0279

Xing B, Wang X, Liu Z (2024) The Wide-Area Coverage Path Planning Strategy for Deep-Sea Mining Vehicle Cluster Based on Deep Reinforcement Learning. Journal of Marine Science and Engineering 12(2): 316. https://doi.org/10.3390/jmse12020316

Xu L, Teoh SS, Ibrahim H (2024). A deep learning approach for electric motor fault diagnosis based on modified InceptionV3. Scientific Reports 14(1): 12344. https://doi.org/10.1038/s41598-024-63086-9

Yeo R (2007) Surveying the underside of an Artic ice ridge using a man-portable GAVIA AUV deployed through the ice. OCEANS, IEEE, 2007: 1-8

Yoshida H, Hyakudome T, Ishibashi S (2013) Yumeiruka-The AUV Equipped with an X-type Canard Rudder. The Twenty-third International Offshore and Polar Engineering Conference 397-401. https://api.semanticscholar.org/CorpusID:115030016

Yuan C, Shuai C, Ma J, Fang Y (2023) An efficient control allocation algorithm for over-actuated AUVs trajectory tracking with fault-tolerant control. Ocean Engineering 273: 113976. https://doi.org/10.1016/j.oceaneng.2023.113976

Žarković M, Stojković Z (2017) Analysis of artificial intelligence expert systems for power transformer condition monitoring and diagnostics. Electric Power Systems Research 149: 125-136. https://doi.org/10.1016/j.epsr.2017.04.025

Zhai JQ, Yang X, Cheng YQ, Li L (2021) A review of the application of machine learning in the field of fault detection and diagnosis. Computer Measurement and Control (3): 1-9. DOI: 10.16526/j.cnki.11-4762/tp.2021.03.001

Zhang MJ, Yin BJ, Liu WX, Wang YJ (2015) Feature extraction and fusion for thruster faults of AUV with random disturbance. Journal of Huazhong University of Science and Technology (Nature Science Edition) 43(6): 22-26, 54. (in Chinese) https://doi.org/10.13245/j.hust.150605

Zhao H, Gao Y, Deng W (2024) Defect Detection Using Shuffle Net-CA-SSD Lightweight Network for Turbine Blades in IoT. IEEE Internet of Things Journal 11(20): 32804-32812. https://doi.org/10.1109/JIOT. 2024.3409823