International Journal of Computer Science and Information Technology
ISSN: 3005-9682 (Print), ISSN: 3005-7140 (Online) | Volume 3, Number 3, Year 2024
DOI: https://doi.org/10.62051/ijcsit.v3n3.17
Journal homepage: https://wepub.org/index.php/IJCSIT/index

Progress and Challenges in Applying Deep Reinforcement Learning to Intelligent Navigation

Yuchi Zhuang *

Nanjing University of Posts and Telecommunications, Nanjing, China
*Corresponding Author: 895206582@qq.com

ABSTRACT

This article mainly discusses the application methods, progress and challenges of deep reinforcement learning (DRL) in intelligent navigation. With the development of computer and artificial intelligence technology, deep learning and reinforcement learning are combined to form deep reinforcement learning. This method shows significant advantages in processing high-dimensional state spaces and complex decision-making tasks. The article first reviews traditional navigation methods, including simulated annealing methods, artificial potential field methods, and fuzzy logic methods. Then it analyzes graph-based methods such as A* algorithm, probabilistic landmark method, and rapid exploration of random trees, as well as bionic intelligence methods such as Genetic algorithm, artificial neural network, ant colony optimization and particle swarm optimization, etc. Subsequently, the article introduces in detail the basic principles of reinforcement learning and its value-oriented, strategy-oriented and combination-oriented methods, focusing on the specific application of deep reinforcement learning in navigation tasks, such as deep Q network (DQN), deep deterministic strategy gradient (DDPG) and dominant actor-critic (A2C) algorithms. Finally, the article discusses the advantages, challenges and future development directions of deep reinforcement learning in navigation applications, emphasizing the technology's path planning and decision-making optimization capabilities in complex dynamic environments.

KEYWORDS

Deep reinforcement learning; Intelligent navigation; Path planning; Deep Q network; Policy gradient

1. INTRODUCTION

1.1. Research Background and Importance

Intelligent navigation systems play a vital role in the development of modern technology, especially in the fields of robotics and autonomous driving. With the advancement of science and technology, traditional navigation methods such as simulated annealing, artificial potential field and fuzzy logic methods perform well in certain application scenarios, but they are insufficient when dealing with high-dimensional state spaces and complex decision-making tasks [1]. In recent years, the combination of deep learning and reinforcement learning, namely Deep Reinforcement Learning (DRL), has become an important tool to solve this problem. DRL enables the system to navigate autonomously in complex and dynamic environments by combining the powerful perception capabilities of deep neural networks and the adaptive decision-making mechanism of reinforcement learning [2].

In the field of intelligent navigation, graph-based methods such as A* algorithm, probabilistic landmark method and rapid exploration random tree (RRT) method, as well as bionic intelligence methods such as genetic algorithm, artificial neural network, ant colony optimization and particle swarm optimization are also widely used. Application [3]. However, these methods still face many challenges when facing dynamic and unknown environments. In contrast, DRL can gradually optimize its decision-making strategy through continuous interaction and learning, thereby improving the efficiency and accuracy of navigation [4]. This advantage makes DRL show great potential in intelligent navigation.

1.2. Research Objectives

The research goal of this article is to comprehensively explore the application methods, progress and challenges of deep reinforcement learning in intelligent navigation. First, we review traditional navigation methods and analyze their advantages, disadvantages and applicable scenarios. Then, the basic principles of reinforcement learning and its value-oriented, strategy-oriented and combination-oriented methods are introduced in detail, and the specific application of deep reinforcement learning in navigation tasks is discussed in detail, such as deep Q network (DQN), deep deterministic policy gradient (DDPG) and the dominant actor-critic (A2C) algorithm. Finally, the advantages, challenges and future development directions of deep reinforcement learning in intelligent navigation applications are discussed, with a view to providing reference and guidance for related research and applications.

2. LITERATURE REVIEW

2.1. Research on Traditional Navigation Methods

Traditional navigation methods play an important role in the field of intelligent navigation, especially when dealing with known environments and static obstacles. These methods include inertial navigation systems (INS), global navigation satellite systems (GNSS), and other sensor-based data fusion technologies. For example, Bird and Arden (2011) studied indoor navigation methods based on foot-mounted small inertial navigation systems and magnetic sensors, which are able to provide precise positioning and navigation information in complex indoor environments despite facing local magnetic field interference, etc. challenge [5]. Erkeç and Haciyev (2019) discussed the relative navigation methods of small satellites, compared different methods of combining inertial navigation systems with laser and inertial navigation systems, and emphasized the advantages and disadvantages of these methods in solving relative navigation problems [6]. The polar grid navigation method proposed by Ping-pin (2015) uses the vertex meridian of the great circle route as a reference. Research has found that this method provides more accurate position information in navigation in high latitudes [7]. In addition, traditional navigation methods such as visual navigation technology are also widely used in actual scenes. By utilizing cameras and other visual sensors, these methods can effectively avoid obstacles and achieve path planning [8].

2.2. Progress in Reinforcement Learning Methods

With the continuous development of machine learning technology, the application of reinforcement learning (RL) in the navigation field has also made significant progress. Reinforcement learning learns optimal strategies through interaction with the environment, enabling autonomous navigation in dynamic and unknown environments. Shao et al. (2018) proposed a visual navigation method based on the dominant actor-critic (A2C) and generalized dominance estimation (GAE) algorithms, which significantly reduces the training time and improves the accuracy of policy gradient estimation. [9]. The memory and knowledge-based asynchronous dominant actor-critic (MK-A3C) algorithm proposed by Zeng et al. (2019) enhances the robot's temporal reasoning ability in complex

environments by constructing a GRU-based memory neural network and utilizes The domain knowledge reward function and transfer learning task architecture solve the non-convergence problem caused by sparse rewards [10]. In addition, Nwaonumah and Samanta (2020) studied the visual navigation problem of wheeled mobile robots (WMR) in dynamic and unknown environments based on deep Q network (DQN) and A3C algorithm. The results showed that the A3C algorithm has better performance under multiple computing threads. The performance is better than the DQN algorithm [11].

2.3. Application of Deep Reinforcement Learning in Navigation

Deep Reinforcement Learning (DRL) shows great potential for autonomous navigation in complex and dynamic environments by combining deep neural networks and reinforcement learning techniques. Chen et al. (2020) proposed a map-based deep reinforcement learning method to train a convolutional neural network (CNN) to predict the correct steering action of the robot by collecting experience in a simulated environment, and successfully applied it to actual mobile robots [12]. Hussein et al. (2017) studied the application of deep imitation learning in three-dimensional navigation tasks. By combining demonstration learning and active learning, this method can effectively complete navigation tasks in unseen environments [13]. Sha et al. (2018) proposed a deep reinforcement learning method based on the visual navigation task, which enabled the agent to successfully navigate in the health collection scene in the ViZDoom environment by training the synchronized dominant actor-critic (A2C) algorithm [14]. These studies show that deep reinforcement learning has significant advantages in handling high-dimensional state spaces and complex decision-making tasks, and can achieve efficient autonomous navigation in a variety of complex environments.

3. CONVENTIONAL NAVIGATION METHOD

Conventional methods can be broadly categorized into three main types: traditional methods, graph-based methods, and bio-inspired intelligent methods:

1) Traditional methods

Traditional methods primarily include Simulated Annealing (SA), Artificial Potential Field (APF), and Fuzzy Logic (FL). These conventional path planning methods are characterized by algorithmic simplicity, strong real-time performance, and wide applicability. They typically have straightforward structures, enabling easy programming, deployment, and real-time application, supporting quick responses to environmental changes and effective navigation in dynamic environments. However, these methods' performance is highly sensitive to parameter settings, requiring careful tuning for optimal results. Despite their advantages, traditional methods face limitations in certain complex scenarios; for instance, the APF method can easily get trapped in local optima, and FL relies heavily on expert knowledge and rule design.

2) Graph-based methods

Graph-based methods mainly include the A* algorithm, Grid-based methods, Probabilistic Roadmaps (PRM), and Rapidly-exploring Random Trees (RRT). Each graph-based method has unique characteristics for path planning. The A* algorithm combines heuristic search with shortest path search, offering high efficiency and optimal solution finding, but can have high memory usage in complex environments. Grid-based methods achieve path planning through simple grid division, making them easy to implement but with accuracy depending on grid resolution and thus suitable for structured environments. Probabilistic Roadmaps (PRM) use random sampling to construct node graphs, which are suitable for high-dimensional spaces and complex environments but require significant initial computation and can get trapped in local optima. Rapidly-exploring Random Trees (RRT) expand tree structures through random sampling, making them suitable for dynamic and

complex environments with high computational efficiency but initial path quality may be low. Each of these methods has pros and cons, and selecting the appropriate method for specific application scenarios can effectively enhance efficiency and accuracy of path planning.

3) Bio-inspired intelligent methods

Bio-inspired intelligent methods primarily include Genetic Algorithms, Artificial Neural Networks, Ant Colony Optimization, Particle Swarm Optimization, and Artificial Bee Colony algorithms. These methods are characterized by their strong global search capability, adaptability, distributed processing, and simplicity of implementation. Genetic Algorithms and Ant Colony Optimization exhibit excellent global optimization abilities in complex environments and high adaptability, albeit with significant computational overhead. Artificial Neural Networks are well-suited for handling nonlinear and high-dimensional data, offering strong learning capabilities, though they require long training times. Particle Swarm Optimization and Artificial Bee Colony algorithms are easy to implement and suitable for real-time optimization problems, but they are sensitive to parameter settings. By emulating biological behaviors observed in nature, these algorithms provide powerful path planning and optimization capabilities, making them suitable for various complex environments and application scenarios. However, they also have drawbacks, such as high computational cost, slow convergence, and the complexity of parameter tuning.

4. REINFORCEMENT LEARNING METHODS

4.1. Basic Principles of Reinforcement Learning

The basic principle of reinforcement learning is that an agent continuously learns through the stimulation of rewards or punishments from environmental feedback. Based on this feedback, the agent continuously adjusts its strategy to maximize the reward or achieve a specific goal. Reinforcement learning methods mainly consist of four elements: state, policy, action, and reward. In state s_t, the agent selects an action a_t according to policy π, transitions from state s_t to a new state s_{t+1}, and receives a reward r from the environment. The agent then uses the received reward r to update and optimize its policy π, aiming to determine the optimal policy π^*.

$$\pi^* = \underset{\pi}{\text{argmax}} E_\pi[\sum_{t=0}^{\infty} \gamma^t r(S_t, a_t) | S_0 = S] \tag{1}$$

Methods for solving reinforcement learning problems can be categorized into value-based, policy-based, and combined value and policy-based approaches. Value-based methods define a value function and select actions based on the value of this function. Policy-based methods parameterize the policy and optimize the parameters to maximize the cumulative return of the policy.

Because of the exploratory nature of initial strategies in navigation tasks within unknown environments, agents often experience slow convergence. Moreover, as the complexity of the environment increases or the system's dimensionality rises, the number of parameters needing training grows exponentially, consuming significant training time and memory, leading to the curse of dimensionality. Furthermore, the poor portability and generalization of reinforcement learning imply that agents trained in one environment cannot be expected to plan movements effectively in a new environment.

4.2. Value-Based Reinforcement Learning Methods

Value-based methods are primarily suited for discrete action spaces, aiming to derive the optimal policy by maximizing the value function for each state. The value function is used to evaluate the effectiveness of the policy chosen by the robot in the current state. Depending on the variables, the

value function can be categorized into the state value function $V(s)$ and the state-action value function $Q(s,a)$, as shown in (2) and (3).

$$V^{\pi}(s) = E_{\pi}[\sum_{t=0}^{\infty} \gamma^{t} r(S_t, a_t) | S_0 = S] \tag{2}$$

$$Q^{\pi}(s_t, a_t) = r(s_t, a_t) + \gamma V^{\pi}(s_{t+1}) \tag{3}$$

The state value function represents the reward feedback for a given state, while the state-action value function represents the reward feedback for a state-action pair. By maximizing these value functions, the ultimate goal of reward maximization can be achieved. Value-based methods primarily include TD (Temporal Difference), Q-Learning, SARSA, among others.

1) TD method

The TD method is a type of model-free reinforcement learning algorithm that samples from the environment and learns an estimate of the current value function. The principle is to update the value function using the temporal difference error. The error calculation formula and the value function update formula are given by (4) and (5), respectively.

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{4}$$

$$V(s_t) \leftarrow V(s_t) + \alpha \delta_t \tag{5}$$

Where α represents the learning rate in the formulas.

The TD method is based on Monte Carlo methods and dynamic programming concepts. It can directly learn from initial experiences without the need for an environmental dynamics model. At the same time, it updates based on learning, requiring waiting for the final learning results.

2) Q-Learning method

Building on the TD method, Watkins proposed the Q-Learning method. The Q-Learning method is a milestone in the development of reinforcement learning. It is the most widely used value-based reinforcement learning algorithm and one of the most effective algorithms currently applied to mobile robot path planning. Q-Learning is an online reinforcement learning algorithm. The basic idea is to define a state-action value function $Q(s,a)$ and update this value function using the data at a given time according to (6) and (7).

$$Q(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t \delta_t \tag{6}$$

$$\delta_t = r_{t+1} + \gamma \max_{a'} Q_t(s_{t+1}, a') - Q(s_t, a_t) \tag{7}$$

Where a_t represents the learning rate, $\delta_t$represents the error, and a' represents the action executed in state s_{t+1}.

The Q-Learning algorithm uses an off-policy strategy to generate actions. It learns the optimal $Q(s,a)$ by interacting with the environment, using the next state and reward obtained from the action taken.

3) SARSA method

The SARSA method is similar to the Q-Learning algorithm and is also an online reinforcement learning algorithm. The key difference is that SARSA uses an on-policy strategy, meaning it updates

the value of $Q(s,a)$ based on the actual actions taken according to the current policy. The error calculation formula for SARSA is given as follows.

$$\delta_t = r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \tag{8}$$

The update of the SARSA value function $Q(s,a)$ involves five components: $(s, a, r, s_{t+1}, a_{t+1})$. These components form the name of the algorithm: SARSA, which stands for State, Action, Reward, State (next), and Action (next).

In machine learning, if an agent learns online and focuses on the rewards obtained during the learning process, the SARSA method is more suitable. The SARSA method is a one-step update algorithm, known as SARSA(0). Upon receiving a reward, it only updates the Q-value corresponding to the previous state and action. However, since the reward obtained at each step can influence the final reward, this algorithm can be optimized into a multi-step update version called SARSA(λ).

The SARSA and Q-Learning methods have similarities and differences. Their similarities include: (1) both are improvements based on the TD method; (2) both use $\varepsilon - greedy$ to select new actions; (3) both are online reinforcement learning algorithms. The differences between these two algorithms are: (1) Q-Learning uses an off-policy approach, iterating on the maximum value of $Q(s,a)$; (2) SARSA uses an on-policy approach, iterating on the actual value of $Q(s,a)$.

4.3. Policy-Based Reinforcement Learning Methods

Policy-based methods achieve the optimal policy by directly optimizing the policy itself. These methods primarily include Policy Gradient (PG) and Imitation Learning (IL).

1) Policy Gradient

The policy gradient method is one of the most fundamental algorithms in policy-based methods. The basic idea is to approximate the optimal policy by directly optimizing the policy. Policy gradient methods can be divided into deterministic policy gradient (DPG) and stochastic policy gradient (SPG). In DPG methods, the action is executed deterministically with a probability of 1, while in SPG methods, actions are executed with certain probabilities. Compared to SPG methods, DPG methods perform better in continuous action spaces.

Assuming the policy to be approximated is $\pi(s,a;\theta)$, where the policy $\pi\pi$ is differentiable with respect to the parameter θ, the objective function and value function are defined as in Equations (9) and (10). Starting from the initial state s_0, the distribution of actions selected according to the policy π_θ is given by Equation (11). The policy gradient formula derived from Equations (9) through (11) is shown in Equation (12).

$$J(\pi_\theta) = E[\sum_{t=1}^{\infty} \gamma^{t-1} r_t | s_0, \pi_\theta] \tag{9}$$

$$Q^{\pi_\theta}(s,a) = E[\sum_{k=1}^{\infty} \gamma^{k-1} r_{t+k} | s_t = s, a_t = a, \pi_\theta] \tag{10}$$

$$d^{\pi_\theta}(s) = \sum_{t=1}^{\infty} \gamma^t P(s_t = s | s_0, \pi_\theta) \tag{11}$$

$$\nabla_\theta J(\pi_\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \nabla_\theta \pi_\theta(s,a) Q^{\pi_\theta}(s,a) \tag{12}$$

2) Imitation Learning

Similar to the policy gradient method, imitation learning is also a direct policy search method [18]. The fundamental principle of imitation learning is to learn from examples provided by a demonstrator, who typically offers decision data from human experts. By mimicking the expert's behavior, the agent derives a policy that approximates the expert's strategy. Under the linear assumption, the feedback

signal can be represented as a linear combination of a set of basis functions $\varphi 1, \ldots, \varphi k$. Therefore, the value of the policy can be expressed as follows:

$$E_{s_0 \sim D}[V^{\pi}(s_0)] = E[\sum_{t=0}^{\infty} \gamma^t \varphi(s_t)|\pi] = E[\sum_{t=0}^{\infty} \gamma^t \omega \varphi(s_t)|\pi] = \omega E[\sum_{t=0}^{\infty} \gamma^t \varphi(s_t)|\pi] \tag{13}$$

If the feature expectations of a policy π satisfy $[\mu(\pi) - \omega^t \mu_E]_2 \leqslant \varepsilon$, as shown in Equation (14), then the policy π is a solution to the imitation learning method.

$$|E[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi_E] - E[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi]| = |\omega^t \mu(\pi) - \omega^t \mu_E| \leqslant |\omega|_2 |\mu(\pi) - \mu_E| \leqslant \varepsilon \tag{14}$$

The above solution process is fundamentally different from direct learning methods that obtain the optimal policy by calculating cumulative reward values. In multi-step decision-making, learning methods based on cumulative rewards often face issues of excessively large search spaces and high computational costs. Imitation learning methods can effectively address these problems in multi-step decision-making.

4.4. Combined Value and Policy-Based Reinforcement Learning Methods

The principle of the Actor-Critic algorithm is as follows: the Actor selects actions based on a probability distribution, and the Critic provides feedback on the selected actions in the form of rewards. The Actor then updates the action selection probabilities based on the Critic's feedback. The parameter update formula for the Actor's policy function is as follows:

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} \log \pi_{\theta}(a \mid s) \delta \tag{15}$$

Where α is the learning rate, $\pi_{\theta}(a \mid s)$ is the policy, and δ is the temporal difference error calculated by the Critic.

To update the Critic's network parameters ω using the mean squared error loss function, the following formula is applied:

$$\omega \leftarrow \omega - \alpha \nabla_{\omega} (\delta_t)^2 \tag{16}$$

Where α is the learning rate and δ_t is the temporal difference error, which can be calculated as:

$$\delta_t = r_t + \gamma V(s_{t+1}; \omega) - V(s_t; \omega) \tag{17}$$

Here, r_t is the reward at time step t, γ is the discount factor, $V(s_{t+1}; \omega)$ is the estimated value of the next state, and $V(s_t; \omega)$ is the estimated value of the current state.

Thus, the Critic's network parameters ω are updated to minimize the mean squared error between the predicted value and the observed return, improving the accuracy of the value function estimation.

5. NAVIGATION BASED ON DEEP REINFORCEMENT LEARNING

The ultimate goal of reinforcement learning is to obtain the optimal policy by maximizing the reward value, thereby exhibiting strong decision-making capabilities. In increasingly complex real-world applications, it is necessary to utilize deep learning to extract high-level features from raw, large-scale data. While deep learning has strong perception capabilities, it lacks certain decision-making

abilities. Deep Reinforcement Learning (DRL) combines the decision-making capabilities of reinforcement learning with the perception capabilities of deep learning, enabling direct control based on input information and resembling human thinking more closely. Path planning techniques based on deep reinforcement learning leverage the power of deep learning and reinforcement learning to navigate complex and dynamic environments, performing effectively in high-dimensional state spaces and learning optimal policies for path planning tasks through interaction with the environment.

5.1. Deep Q-Network

In 2013, Google's AI research team DeepMind [21] innovatively combined the Q-Learning algorithm with convolutional neural networks, introducing the Deep Q-Network (DQN). The foundational model of DQN is a convolutional neural network trained using a variant of Q-Learning. DQN made the following key improvements to the traditional Q-Learning algorithm:

1) Replacing the State-Action Value Function $Q(s,a)$ with a Convolutional Neural Network: The value function $Q(s,a;\theta_i)$ with parameters θ_i is used, and the loss function after i iterations is expressed as follows:

$$L_i(\theta_i) = E_{s,a,r,s'}[(Y_i - Q(s,a;\theta_i))^2] \tag{18}$$

The term Y_i approximates the optimization target of the value function. The calculation formula for Y_i is as follows:

$$Y_i = r + \gamma \max_{a'} Q(s',a;\theta^-) \tag{19}$$

During the learning process, θ_i is updated to θ^-. The specific learning process involves taking the derivative of the loss function with respect to θ_i to obtain the gradient:

$$\begin{aligned}\nabla_{\theta_i} L_i(\theta_i) = E_{s,a,r,s'}[(r + \gamma \max_{a'} Q(s',a;\theta^-) - \\ Q(s,a;\theta_i))\nabla_{\theta_i} Q(s,a;\theta_i)]\end{aligned} \tag{20}$$

2) Using Experience Replay: At each time step t, store the agent's experience sample $et = (s_t, a_t, r_t, s_{t+1})$ into the replay memory $D = \{e_1, e_2, \ldots, e_t\}$. By repeatedly sampling from historical data, experience replay increases the sample efficiency and effectively avoids oscillations in the parameter updates during learning.

3) Sampling Random Mini-Batches from Replay Memory D: Due to the high correlation between consecutive samples, learning directly from sequential samples is inefficient. By sampling random mini-batches from the replay memory, the correlation between samples is reduced, thereby enhancing the stability of the algorithm.

5.2. Deep Deterministic Policy Gradient

The Deep Deterministic Policy Gradient (DDPG) algorithm was developed to address the challenges of reinforcement learning in continuous action spaces [22]. Traditional reinforcement learning algorithms like Q-Learning and SARSA were primarily designed for discrete action spaces, making them less effective in domains where actions need to be selected from a continuous range. The development of DDPG was influenced by the need for more robust and scalable solutions in applications such as robotics, autonomous driving, and complex control systems.

1) Actor-Critic Architecture

DDPG uses completely separate deep neural networks for the actor and critic, allowing each to specialize and optimize independently.

Actor network ($\mu(s \mid \theta^{\mu})$) determines the action a to take given a state s. It outputs deterministic actions directly from the continuous action space. And critic network ($Q(s, a \mid \theta^{Q})$) evaluates the action taken by the actor by estimating the Q-value, which is the expected return of taking action a in state s.

Then DDPG employs target networks for both the actor and critic. These target networks are slowly updated copies of the original networks, providing stable targets for learning.

The parameters of the target networks $\theta^{\mu'}$ and $\theta^{Q'}$ are updated using soft updates:

$$\begin{aligned} \theta^{\mu'} &\leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \\ \theta^{Q'} &\leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'} \end{aligned} \tag{21}$$

Where $\tau \ll 1$ is the update rate.

2) Experience Replay Buffer

Traditional Actor-Critic issues learn from consecutive experiences. And it can introduce correlations that slow down learning and lead to suboptimal policies. DDPG uses an experience replay buffer to store transitions (s_t, a_t, r_t, s_{t+1}). During training, mini-batches of experiences are randomly sampled from this buffer.

This approach reduces the correlation between samples and improves sample efficiency, leading to more stable and faster learning.

3) Policy and Value Function Updates

Critic Update:

The critic network is updated by minimizing the loss function:

$$L(\theta^{Q}) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1})\sim D}[(y_t - Q(s_t, a_t|\theta^{Q}))^2] \tag{22}$$

Where the target value y_t is computed using the target networks:

$$y_t = r_t + \gamma Q'(s_{t+1}, \mu'(s_{t+1}|\theta^{\mu'})|\theta^{Q'}) \tag{23}$$

Here, r is the reward received after taking action a_t in state s_t, γ is the discount factor, and Q' and μ' are the target networks for the critic and actor, respectively.

Actor Update:

The actor network is updated by maximizing the expected return, using the deterministic policy gradient theorem:

$$\nabla_{\theta^{\mu}} J \approx \mathbb{E}_{s_t\sim D}\left[\nabla_{\theta^{\mu}} Q(s, a|\theta^{Q})|_{a=\mu(s|\theta^{\mu})}\right] \tag{24}$$

Applying the chain rule for gradients, this becomes:

$$\nabla_{\theta^{\mu}} J \approx \mathbb{E}_{s_t\sim D}\left[\nabla_{a} Q(s, a|\theta^{Q})|_{a=\mu(s|\theta^{\mu})} \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})\right] \tag{25}$$

4) Exploration Strategy

Deterministic policies can lead to insufficient exploration, especially in continuous action spaces. DDPG adds exploration noise to the deterministic policy output to encourage exploration.

Typically, an Ornstein-Uhlenbeck process is used to generate temporally correlated noise suitable for physical control problems:

$$a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t \tag{26}$$

Where $\mathcal{N}_t$ is the noise added for exploration.

5.3. Advantage Actor-Critic

Advantage Actor-Critic (A2C) is an improved version of the traditional actor-critic methods designed to enhance stability and efficiency. It addresses some of the limitations of previous approaches and introduces several key improvements. Here's a detailed description of the improvements made by A2C:

1) Synchronous Parallelization

A2C runs multiple instances of the environment in parallel, collecting experiences from multiple agents simultaneously. This synchronous parallelization leads to better utilization of computational resources and faster training. This parallel approach helps to stabilize training by averaging out the noise over multiple agents, leading to more robust learning updates.

2) Advantage Function

A2C uses the advantage function to reduce variance and improve learning efficiency. The advantage function $A(s,a)$ is defined as the difference between the Q-value and the value function:

$$A(s,a) = Q(s,a) - V(s) \tag{27}$$

This represents how much better or worse an action is compared to the average action taken in that state.

3) Objective Function

A2C updates the objective function to incorporate the advantage function, leading to more stable updates. The policy gradient update rule in A2C is:

$$J = \mathbb{E}_t[\log \pi_\theta(a_t|s_t)A_t + \beta\mathcal{H}(\pi_\theta(\cdot\,|s_t))] \tag{28}$$

Where $H(\pi_\theta)$ is the entropy of the policy and β is a hyperparameter that controls the strength of the entropy regularization.

6. CHALLENGES OF DEEP REINFORCEMENT LEARNING IN NAVIGATION

6.1. Parameter Training Challenges Brought by Increasing Environmental Complexity and System Dimensions

As the complexity of the navigation environment and system dimensions increase, deep reinforcement learning (DRL) faces huge challenges in parameter training. Complex environments and high-dimensional state spaces will significantly increase the difficulty of training DRL models, requiring more computing resources and time to adjust a large number of parameters. Kulhánek et al. (2019) proposed that in the face of complex visual navigation tasks, traditional DRL models are difficult to converge quickly in highly complex environments, and it is necessary to improve the training performance of the model by introducing auxiliary tasks and gradually increasing the

complexity of the environment [15]. In addition, Zhu et al. (2021) pointed out that navigation tasks in complex environments usually need to deal with a large amount of uncertainty and dynamic changes, which further increases the difficulty of parameter adjustment and the complexity of training [16]. These studies show that in the face of high-complexity environments, the training of DRL models requires more effective strategies and stronger computing power.

6.2. Problems with Portability and Generalization Ability of Deep Reinforcement Learning

The portability and generalization ability of deep reinforcement learning models in different environments is another important challenge. Most current DRL models perform well in specific environments, but when faced with new, unseen environments, the models often need to be retrained or fine-tuned, resulting in limited flexibility in practical applications. Dhiman et al. (2018) found that although DRL algorithms can effectively utilize map information in the same training environment, when the test environment is different from the training environment, the performance of these algorithms decreases significantly and the generalization ability is insufficient [17]. Devo et al. (2020) also pointed out that current DRL methods need to be retrained for each new environment and goal in goal-driven visual navigation, which is very challenging and risky in practical operations [18]. These problems highlight the limitations of DRL in practical applications and require the development of generalization strategies that can better adapt to new environments and goals.

6.3. Training Time and Memory Consumption in High-Dimensional State Space

Training time and memory consumption in high-dimensional state spaces are another major challenge for deep reinforcement learning. In a high-dimensional state space, the DRL model needs to process a large amount of data, which not only increases the training time, but also places higher requirements on computing resources and memory. Lin et al.'s (2021) research shows that in complex and large-scale environments, DRL algorithms need to reduce training time and memory consumption through improved strategies and reward mechanisms [19]. Zheng et al. (2023) proposed that by converting the interactive environment state representation, the computational complexity and memory usage of the model can be reduced, thereby improving the training efficiency and the convergence speed of the model [20]. These studies highlight that optimizing training time and memory consumption is crucial for the application of DRL models in high-dimensional state spaces.

7. CONCLUSION AND FUTURE WORK

7.1. Main Conclusions

This article systematically reviews the application methods of deep reinforcement learning in intelligent navigation, and analyzes its basic principles, characteristics, advantages and disadvantages, and applicable occasions. Through a detailed elaboration of conventional navigation methods, reinforcement learning methods, and deep reinforcement learning methods, the significant advantages of deep reinforcement learning in processing high-dimensional state spaces and complex decision-making tasks are demonstrated. In particular, the successful application of algorithms such as Deep Q Network (DQN), Deep Deterministic Policy Gradient (DDPG) and Advantaged Actor-Critic (A2C) in path planning demonstrates the great potential of deep reinforcement learning in complex dynamic environments. In general, deep reinforcement learning provides new solutions for navigation tasks and is of great significance to promoting the development of path planning technology.

7.2. Future Research Directions

Future research should focus on improving the efficiency and generalization ability of deep reinforcement learning algorithms. First of all, the network structure and training algorithm can be optimized to reduce training time and memory consumption and improve the efficiency of the algorithm. Secondly, it is an important research direction to enhance the portability and generalization ability of the algorithm so that it can perform well in different environments. In addition, combined with other artificial intelligence technologies, such as transfer learning, multi-agent learning, etc., the performance in navigation tasks can be further improved. Finally, apply deep reinforcement learning to more practical scenarios, such as driverless driving, robot path planning, etc., to promote its widespread application in industry and life. Through these efforts, the potential of deep reinforcement learning in intelligent navigation will be further explored and realized.

REFERENCES

[1] Kulhánek, J., Derner, E., Bruin, T. D., Babuška, R.: Vision-based Navigation Using Deep Reinforcement Learning. 2019 European Conference on Mobile Robots (ECMR), pp. 1-8 (2019).

[2] Choi, J., Dance, C., Kim, J., Park, K., Han, J., Seo, J., Kim, M.: Fast Adaptation of Deep Reinforcement Learning-Based Navigation Skills to Human Preference. 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 3363-3370 (2020).

[3] Santos, I. B. D. A., Romero, R.: Deep Reinforcement Learning for Visual Semantic Navigation with Memory. 2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE), pp. 1-6 (2020).

[4] Shi, H., Shi, L., Xu, M., Hwang, K.: End-to-End Navigation Strategy With Deep Reinforcement Learning for Mobile Robots. IEEE Transactions on Industrial Informatics, 16, pp. 2393-2402 (2020).

[5] Bird, J., Arden, D.: Indoor navigation with foot-mounted strapdown inertial navigation and magnetic sensors [Emerging Opportunities for Localization and Tracking]. IEEE Wireless Communications, 18, pp. 28-35 (2011).

[6] Erkeç, T. Y., Haciyev, C.: Traditional Methods on Relative Navigation of Small Satellites. 2019 9th International Conference on Recent Advances in Space Technologies (RAST), pp. 869-874 (2019).

[7] Ping-pin, Z.: Research on Polar Grid Navigation of Great-Circle Sailing. Command Control & Simulation (2015).

[8] Forsman, F., Sjörs-Dahlman, A., Dahlman, J., Falkmer, T., Lee, H. C.: Eye Tracking During High Speed Navigation at Sea. Journal of Transportation Technologies, 2, pp. 277-283 (2012).

[9] Shao, K., Zhao, D., Zhu, Y., Zhang, Q.: Visual Navigation with Actor-Critic Deep Reinforcement Learning. 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1-6 (2018).

[10] Zeng, J., Ju, R., Qin, L., Hu, Y., Yin, Q., Hu, C.: Navigation in Unknown Dynamic Environments Based on Deep Reinforcement Learning. Sensors (Basel, Switzerland), 19 (2019).

[11] Nwaonumah, E. M., Samanta, B.: Deep Reinforcement Learning for Visual Navigation of Wheeled Mobile Robots. 2020 SoutheastCon, pp. 1-8 (2020).

[12] Chen, G., Pan, L., Chen, Y., Xu, P., Wang, Z., Wu, P., Ji, J., Chen, X.: Robot Navigation with Map-Based Deep Reinforcement Learning. 2020 IEEE International Conference on Networking, Sensing and Control (ICNSC), pp. 1-6 (2020).

[13] Hussein, A., Elyan, E., Gaber, M., Jayne, C.: Deep imitation learning for 3D navigation tasks. Neural Computing and Applications, 29, pp. 389-404 (2017).

[14] Shao, K., Zhao, D., Zhu, Y., Zhang, Q.: Visual Navigation with Actor-Critic Deep Reinforcement Learning. 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1-6 (2018).

[15] Kulhánek, J., Derner, E., Bruin, T. D., Babuška, R.: Vision-based Navigation Using Deep Reinforcement Learning. 2019 European Conference on Mobile Robots (ECMR), pp. 1-8 (2019).

[16] Zhu, K., Zhang, T.: Deep reinforcement learning based mobile robot navigation: A review. Tsinghua Science & Technology, 26, pp. 674-691 (2021).

[17] Dhiman, V., Banerjee, S., Griffin, B. A., Siskind, J., Corso, J. J.: A Critical Investigation of Deep Reinforcement Learning for Navigation. ArXiv (2018).

[18] Devo, A., Mezzetti, G., Costante, G., Fravolini, M., Valigi, P.: Towards Generalization in Target-Driven Visual Navigation by Using Deep Reinforcement Learning. IEEE Transactions on Robotics, 36, pp. 1546-1561 (2020).

[19] Lin, C., Hasan, S., Bai, O.: Robotic Navigation with Human Brain Signals and Deep Reinforcement Learning. 2021 4th International Conference on Robotics, Control and Automation Engineering (RCAE), pp. 278-283 (2021).

[20] Zheng, Q., Huang, X., Dong, C., Liu, Y., Chen, D.: An improved deep reinforcement learning for robot navigation. (2023).