

Received 26 April 2024, accepted 5 May 2024, date of publication 13 May 2024, date of current version 20 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3400159

RESEARCH ARTICLE

An Improved Deep Reinforcement Learning Algorithm for Path Planning in Unmanned Driving

KAI YANG¹ AND LI LIU²

¹School of Mathematics and Information Science, Nanjing Normal University of Special Education, Nanjing 210038, China

²School of Science, Hainan University, Haikou 570228, China

Corresponding author: Li Liu (lliu092@126.com)


This work was supported in part by the Incorporation Project of Nanjing under Grant 71020125365H.

ABSTRACT In the domain of intelligent transportation systems, the advent of autonomous driving technology represents a critical milestone, profoundly shaping the automotive industry's evolutionary path. This technology's core, particularly the algorithms facilitating driverless path planning, has attracted significant scholarly interest. This paper presents an advanced Deep Reinforcement Learning algorithm for Path Planning (DRL-PP), designed to rectify the shortcomings inherent in existing path planning techniques. Considering the complex nature of the environment, the DRL-PP algorithm is meticulously crafted to ascertain optimal actions, thereby effectively reducing the propensity for overfitting. The algorithm harnesses the capabilities of deep reinforcement learning, utilizing neural networks to identify the most advantageous action corresponding to a specific state. It then constructs an optimal action sequence, extending from the vehicle's initial position to its designated target. Additionally, the algorithm enhances the reward function by incorporating data pertinent to the objective. This refinement enables the nuanced differentiation of action values based on dynamically adjusted reward metrics, thereby augmenting the efficiency of the action selection process and yielding improved results in path planning. Empirical results validate the algorithm's proficiency in stabilizing the reward metric while minimizing exploratory steps, consistently surpassing comparative models in path-finding effectiveness.

INDEX TERMS Deep reinforcement learning, path planning, autonomous driving, deep Q-learning network.

I. INTRODUCTION

In the contemporary era, the increasing incidence of traffic incidents has cast a spotlight on the deficiencies and frailties inherent in manual driving operations [1]. The imperatives of sustained attentiveness in human-driven vehicular navigation, coupled with inherent human limitations such as susceptibility to fatigue and distraction, accentuate these vulnerabilities. The advent of autonomous driving technologies promises a paradigm shift, mitigating the tedium associated with extended periods behind the wheel while simultaneously instituting advanced [2], intelligent control systems that are instrumental in bolstering vehicular safety and operational efficiency.

The associate editor coordinating the review of this manuscript and approving it for publication was Jiachen Yang .

Numerous researchers have delved into autonomous vehicular technology, culminating in a corpus of research that spans a gamut of disciplines [3]. A salient challenge that contemporary autonomous vehicles contend with is the extension of their reactive capabilities and their adaptability in specialized vehicular scenarios. Conventional intelligent driving systems predominantly leverage extensive datasets to inculcate driving behaviors, yet the aggregation of exhaustive data encompassing the full spectrum of road conditions remains an arduous undertaking [4], [5]. While simulation in naturalistic environments offers a partial remedy, the requisite investment in terms of cost and resources is not insignificant. Furthermore, during the initial phase of reinforcement learning, autonomous vehicles—bereft of any pre-existing environmental knowledge—must undertake a thorough exploration of the state space to discern space values and subsequently cultivate optimal driving behaviors [6], [7].

Deep Reinforcement Learning (DRL) marries the perceptual acuity of deep learning with the strategic prowess of reinforcement learning, heralding a vanguard of applications across varied sectors [8]. DRL's foray into the domain of path planning is particularly noteworthy, where it demonstrates remarkable competence in negotiating complex environments and executing challenging tasks with notable alacrity. The Deep Q-Network (DQN) paradigm [9], [10], [11], an amalgamation of Q-learning and deep learning, epitomizes this synergy by transforming the state-action value function into a construct amenable to neural network interpretation, thereby enabling the calculation of action values pertinent to the current state and facilitating the identification of an optimal navigational strategy.

In this paper, we introduce a planning algorithm that is embedded within the DRL framework. Harnessing the predictive capabilities of neural networks, our algorithm ascertains and enumerates superior actions pertinent to discrete states, thereby sculpting an action set that maps a trajectory from origin to terminus with heightened precision. The caliber of the neural network and the refinement of its parameters are of quintessential importance to the process of action selection [12]. Accordingly, we utilize a multi-layer perceptron (MLP), which undergoes parameter optimization through a meticulously calibrated reward function, with the aim of amplifying the efficacy of the action selection mechanism and, by extension, enriching the efficiency of the path-planning paradigm.

The subsequent sections of this paper are organized as follows: Section II elucidates related works, Section III delves into the intricacies of the deep Q-network and deep reinforcement learning, Section IV details the DRL-PP algorithm, Section V presents the experimental design and results, and finally, Section VI provides the concluding remarks.

II. RELATED WORK

Path planning articulates a sequence of decisions for vehicles to navigate future spatial and temporal domains. Typically, it can be bifurcated into local (dynamic) and global (static) path planning based on the scope and temporal duration involved. Local path planning dynamically alters the driving status in response to environmental stimuli, encompassing actions like lane changing and obstacle avoidance. In contrast, global path planning designs a route to a destination based on the vehicle's present position on a pre-established map. The general path-planning procedure can be segregated into three stages:

- Establishing the environment model: Develop an abstracted environment model rooted in the actual road conditions, furnishing a schematic for algorithmic strategies, thereby facilitating computational path planning.
- Path search: Harness the abstract environment model to delineate a route from the inception to the destination, striving to discern the most optimal path.

- Path optimization: Refine the pre-established path by pruning superfluous nodes and augmenting its fluidity.

Recent progress in the field of path planning has increasingly emphasized the adoption of reinforcement learning (RL) techniques. Distinct from traditional machine learning algorithms, RL agents possess the significant advantage of requiring minimal datasets for effective training. This advantage is a pivotal reason for their integration into path planning methodologies. Within this realm, Q-learning (QL) and Deep Q-Learning (DQL) stand out as two predominant RL algorithms. QL, for instance, determines the Q-value using the Bellman equation, as demonstrated in [13]. Nevertheless, QL's scalability is limited, particularly as environmental complexity and memory requirements increase.

Mnih et al. [14] identified that traditional RL struggles with tasks featuring large state spaces and continuous action spaces that resemble real-world complexities. Deep learning (DL), on the other hand, adeptly handles high-dimensional challenges. DRL merges DL's capacity for high-dimensional perception with RL's decision-making abilities. This integration is achieved via neural networks, offering solutions to the dimensional complexities that traditional RL encounters. In this context, DL paves the way for addressing cognitive decisions in complex environments.

Conversely, DQL utilizes neural networks to predict the Q-value, enhancing scalability. The implementation of DQL in grid-world environments for path planning is evidenced in studies like [15] and [16], where RL agents efficiently navigate around static obstacles using available environmental data. The advancement of DQL into dynamic environments, characterized by moving and uncertain obstacles, is explored in [17]. An alternative strategy in unknown environments, as proposed in [18], involves training RL agents with sensor data, offering the potential for dynamic scenario adaptation.

The convergence time, a critical factor in machine learning and especially relevant in path planning, is the period required for an agent to learn a specific task. Efforts to reduce this convergence time in DRL-based path planning algorithms have been substantial. In [19], initial training in a 2D simulator environment, followed by subsequent training in 3D environments, has been shown to reduce convergence time. The integration of generalized knowledge about goals and current states in a Q-learning system, as discussed in [20], significantly decreases training time. Additionally, [21] presents the combination of RL with particle swarm optimization to accelerate the agent's convergence rate.

Deep Reinforcement Learning is fundamentally bifurcated into two categories. Value-based DRL, exemplified by the pioneering work of Mnih et al. with the advent of the Deep Q-Network, encapsulates the process of distilling a value model from sequential informational inputs to inform policy updates [6]. Advancements in this domain include the Double Deep Q-Network (DDQN) by Hasselt et al., which refines the value estimation process of DQN, thereby enhancing

the model's efficacy in navigating complex decision-making tasks [22]. A subsequent innovation, the Dueling DQN conceived by Wang et al., partitions the Q network into distinct streams: one assessing the value of states and the other evaluating the advantages of actions, resulting in more precise Q-value determinations [21].

A novel contribution to this lineage is the quantum-inspired experience replay (QiER) introduced by Li et al., which augments traditional DRL paradigms by incorporating quantum bit (qubit) transition significance and utilizing Grover's iteration for amplitude amplification. This QiER method adeptly balances the dichotomy between sampling priority and diversity, as evidenced by comparative analyses that underscore its preeminence over conventional DRL methodologies and non-learning baselines [23]. This quantum-computational synergy heralds a significant stride in UAV navigational algorithms, encapsulating a unique action selection policy and reinforcement strategy inspired by quantum mechanics' inherent properties—specifically, the collapse phenomenon and amplitude amplification. These principles confer upon the QiRL method an innate equilibrium between exploration and exploitation, alleviating the dependency on meticulously tuned exploration parameters that often encumber traditional reinforcement learning frameworks [24].

The alternate strand of DRL, predicated on policy gradients, is characterized by the iterative refinement of parameters. This is achieved through the persistent computation of gradients concerning the expected total rewards of policies. An exemplary model of this class is Lillicrap's Deep Deterministic Policy Gradient (DDPG), situated within the Actor-Critic framework. It builds upon DQN's foundation—utilizing experience replay and delineated target networks—and adeptly extends its application to continuous action domains [25]. This gradient-based strategy underscores a continuous evolution in DRL, empowering agents to perform with heightened sophistication and adaptability across diverse behavioral spectrums.

Yet, these reinforcement learning methods are primarily designed for single-agent scenarios and are less effective in Multi-Agent Systems (MAS). Lowe et al.'s Multi-Agent DDPG (MADDPG) addresses this shortfall by extending DDPG for global cooperation in MAS. MADDPG employs centralized training and decentralized execution, where a centralized critic module during training offers insights into the observations and potential actions of all agents, making unpredictable environments more predictable. In testing, agents operate independently, guided only by their critic modules. Thus, MADDPG, with its actor-critic network that includes inputs from all agents, effectively mitigates the non-static nature of the environment and is particularly suited for collaborative crowd path planning.

III. DEEP Q-NETWORK

This section elucidates the frameworks integral to DRL-PP, encompassing reinforcement learning and the deep Q-network.

A. REINFORCEMENT LEARNING

At its core, deep reinforcement learning is a derivative of reinforcement learning. It inherently seeks to modify one's action blueprint through sustained interaction with the environment, obtaining rewards reflective of environmental responses post-action [12], [26], [27]. This learning paradigm can be mathematically represented as a Markov decision process, characterized by a quintuple, $\langle S, A, P, r, \gamma \rangle$. Herein, the decision-making process is delineated in Fig 1.

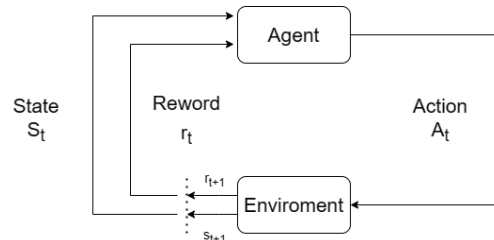


FIGURE 1. Reinforcement learning framework.

Traditional reinforcement learning algorithms, however, find their efficacy diminished in high-dimensional continuous state and action spaces. To address this limitation, enhanced solutions like deep neural network Q-learning algorithms have been devised.

B. NEURAL NETWORK

The Deep Q-Network amalgamates the Q-learning algorithm with deep learning [28], [29], [30]. In this synthesis, a neural network personifies the state-action value function intrinsic to Q-learning. This network ingests the state as its input, producing the value of potential actions in the present state. Consequently, the optimal action for execution is selected based on these derived values.

The DQN algorithm typifies a genre of deep reinforcement learning algorithms. Its crux lies in juxtaposing neural networks with the Q-learning algorithm [31], [32]. By leveraging the formidable feature extraction capabilities of neural networks, it processes sequential images, deeming each a state for reinforcement learning, thereby guiding the subsequent neural network model. This model, in turn, dispenses the Q-value of every conceivable action, facilitating the determination of the most auspicious action in the prevailing state.

C. APPROXIMATING THE VALUE FUNCTION

For low-dimensional, discrete state and action spaces, tabular methods offer a viable solution. However, in scenarios with high-dimensional, continuous state and action spaces, approximation methods come to the fore. Here, the action-value function $Q(s, a, \theta)$ approaches the optimal action-value function $Q^*(s, a)$ via the parameter θ , as expressed in Eq. 1.

$$Q(s, a, \theta) \approx Q^*(s, a). \quad (1)$$

Deep learning’s capacity to autonomously extract intricate features from data makes it a preferred choice for reinforcement learning tasks characterized by continuous high-dimensional state spaces. As a result, deep learning is deployed to construct a value network to discern the value function, drawing on the principles of reinforcement learning.

The efficacy of the DQN algorithm in seamlessly integrating deep learning with reinforcement learning can be attributed to three pivotal innovations:

- Objective function formulation: Rooted in the Q-learning algorithm, rewards are utilized to craft labels, yielding a viable objective function for deep learning.
- Experience Replay: By introducing an experience pool, challenges arising from data correlation and non-stationary distribution are surmounted.
- Dual networks: Two distinct neural networks are employed—one to generate the prevailing Q-value and another for the target Q-value—enhancing model stability.

IV. ALGORITHMIC FRAMEWORK

A. FORMULATION OF THE DRL-PP ALGORITHM

When transforming environmental information into a grid-based format for path planning, the integration of convolutional neural networks with the deep Q-network algorithm can introduce overfitting challenges. This study introduces a refined deep Q-network path planning algorithm that incorporates a multilayer perceptron into the deep Q-network paradigm and employs an enhanced reward function during each neural network iteration.

1) MULTILAYER PERCEPTRON

The Multilayer Perceptron is a neural network architecture featuring multiple hidden layers between its input and output layers. It utilizes backpropagation with gradient descent for training, aiming to minimize classification error through an optimal parameter set ω . Forward propagation in MLP produces output y , represented as:

$$y_j = \sum_{i=0}^M \omega_{ij}x_j, \quad j = 1, 2, \dots, n. \quad (2)$$

Here, i and j denote the neuron indices in the preceding and current layers, respectively, with ω_{ij} as the corresponding weight. The MLP’s design significantly influences the structure of networks within the DRL-PP algorithm, ensuring consistency across network architectures.

For the prediction network, the input is derived from the experience pool (s, a) , where s represents spatial coordinates (x, y) of the object. The network employs a tri-layered fully connected structure to determine the reward for executing action a in state s . Conversely, the target network’s input, sourced from the experience pool (s', r) , involves computing the cumulative reward based on actions in state s' and the corresponding reward r .

2) REWARD MECHANISM AND ACTION SELECTION STRATEGY

In contemporary path planning research, the design of reward structures plays a crucial role, especially when employing reinforcement learning techniques. Traditional reward schemes often adopt a uniform approach, assigning similar reward values to a range of actions within an exploratory path context. This uniformity can undermine the learning process, blurring the distinction between optimal and less effective actions, which may lead to subpar neural network training outcomes.

To counteract this, this paper incorporates advanced principles derived from cubic processing to refine the reward function. This innovation enables the allocation of distinct reward values for different actions, facilitating a more nuanced and effective alignment with the optimal action-state value function. Such an approach ensures a more sophisticated and tailored learning process, crucial for navigating complex path planning scenarios.

Table 1 details the specific reward metrics assigned to different states encountered during the path planning process. ‘Value 1’ is allocated for reaching the target state successfully, thereby reinforcing goal-oriented navigation. Conversely, ‘Value 2’ is assigned as a penalty for colliding with obstacles, thereby encouraging obstacle avoidance strategies.

TABLE 1. Reward allocation table.

Reward Value	State of the Object
Value 1	Successfully Reaching the Goal
Value 2	Encountering an Obstacle
r	Safe Navigation without Obstacle Interaction

The reward function, denoted as R , computes the expected reward for a given state s , as outlined in the table. This computation can be mathematically expressed as follows:

$$R_s = E [R_{t+1} | s = s_t] \quad (3)$$

Here, E denotes the mathematical expectation. The formulation is based on the premise that if the agent successfully navigates without colliding with obstacles, the reward calculation is influenced by whether a target is detected or not:

$$r = \begin{cases} 0, & \text{if no target is detected} \\ l \times (x-h)^3, & \text{if a target is detected} \end{cases} \quad (4)$$

where l is employed to delineate the function’s monotonicity, directly influencing the rapidity with which the function’s output converges towards the symmetry center. Concurrently, the parameter h signifies the x-axis intercept, serving as a pivotal reference point in the function’s graphical representation. Additionally, x represents the Euclidean distance between the agent and its target during the exploration phase, excluding scenarios where the agent is terminating the path or directly finding the target. This measure is crucial for evaluating the

agent’s proximity to its objective throughout the navigation process.

In this framework, R_s encapsulates the projected cumulative reward from a specific time t onward. The action value for the current state is further elucidated in Eq. 5:

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{i=0}^{\infty} \gamma^i R_{t+i+1}. \quad (5)$$

This equation incorporates the immediate reward R_{t+1} and the discount factor γ for future rewards.

For action selection, our model incorporates a strategic balance between exploration and exploitation. The ϵ -greedy strategy is employed, which is a well-recognized method in the realm of RL. This algorithm sets the probability ϵ for choosing the most effective action known to date, while assigning a complementary probability $(1-\epsilon)$ for exploring new actions. Importantly, as the training advances, ϵ gradually increases in line with the number of training iterations. This increment in ϵ strategically reduces random exploration, progressively steering the algorithm towards identifying and selecting the most advantageous actions, thus guiding the learning process towards an optimal path planning solution.

B. NEURAL NETWORK TRAINING

To mitigate undue fluctuations in loss values and enhance the robustness of the neural model, our algorithm implements a dual neural network architecture, comprising a primary network and a target network. This design is pivotal in stabilizing the learning process, as these networks are responsible for generating the current Q value and the target Q value, respectively. This is outlined in Eq. 1, where $Q(s, a, \theta)$ represents the current Q value, and $r + \gamma \max Q(s', a', \theta')$ signifies the target Q value, crucial for determining the direction of model updates. The target Q value calculation is detailed in Eq. 6.

$$\text{Target } Q = \begin{cases} r, & \text{if the goal is reached} \\ r + \gamma \max Q(s', a', \theta'), & \text{otherwise} \end{cases} \quad (6)$$

In Eq. 6, the condition of reaching the goal (or target frame) is indicated by G . The discount factor γ is applied to future rewards, balancing immediate and long-term rewards.

Fig 2 provides a schematic overview of the neural network training logic. Key functions include:

- (1). *Observe*: This function transforms the current state s into a vector format, preparing it as input for the neural network.
- (2). *Predict*: Representing the forward propagation in the neural network, this function calculates the current Q value based on the input state.
- (3). *Fit*: Central to training, this function utilizes parameters *input* (the input vector) and *target* (the desired output). This process adapts the network’s parameters to reduce the difference between predicted and target Q values.

Though the schematic may depict a single data sample, the training process typically involves a mini-batch approach. Each batch, usually containing around 16 samples, allows the network to learn from a variety of scenarios concurrently, thereby enhancing the generalization and efficiency of the learning process. This batched approach is instrumental in facilitating more stable and reliable gradient estimations, a critical aspect of training deep neural networks effectively.

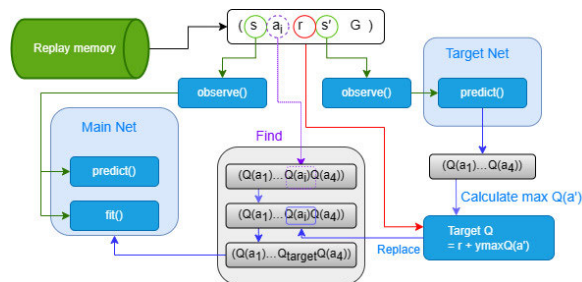


FIGURE 2. DRL-PP update process.

C. ANALYSIS OF ALGORITHM

In this paper, the analysis of computational complexity encompasses both time and space dimensions, each bearing substantial weight in algorithmic efficacy and efficiency. Time complexity in this context pivots largely on the neural network’s architecture, entailing the forward and backward pass computations across its layers. In this paper, a network comprising two hidden layers, each with 128 neurons, engages in computational processes that scale with the product of the inputs and neurons for each layer, rendering a time complexity of $O(n \times m)$ per layer. Moreover, the magnitude of the action space directly influences the computation time for determining the optimal action.

Space complexity, conversely, is anchored in the storage requisites for the neural network’s parameters—weights and biases—as well as the experience replay buffer, integral to the learning process. The storage demand is proportional to the total number of network parameters, which grows with the depth and breadth of the network, alongside the replay buffer capacity, which stores a multitude of agent experiences. The architecture consisting of two sequential layers with 128 neurons each, demands storage for 128^2 weights per layer and an additional 128 biases, amounting to an aggregate of approximately $2 \times (128^2 + 128)$ space units.

The interplay between performance, time, and space complexity manifests as a series of trade-offs. Augmenting the network’s complexity can amplify learning and generalization capabilities (performance) but at the cost of heightened computational time and increased memory requirements. Similarly, enlarging the experience replay buffer enriches the learning landscape but incurs additional space complexity due to escalated memory demands. This intricate balance between the algorithm’s structural intricacies and its operational efficacy delineates the nuanced landscape of

computational complexity in deep reinforcement learning models like DRL-PP.

V. EVALUATIONS

In this section, we systematically assess the efficacy of the algorithm we have put forth. Initially, we delineate the experimental parameters that form the backbone of our investigation. This is succeeded by an elaborate discourse on the evaluation methodology, which is designed to rigorously scrutinize the algorithm's performance against a spectrum of criteria.

A. EXPERIMENTAL SETUP

Considering the inherent challenges of conducting real-world path-planning trials with autonomous vehicles, our study adopted a more pragmatic and controlled approach: we validated the path-planning algorithms through agent trajectory simulations.

For our experimental setup, we utilized the TensorFlow deep learning platform, a choice influenced by its robust computational graph architecture, which efficiently facilitates neural network training and inference. Our computational environment was based on the reliable and widely-used Ubuntu operating system, known for its performance and compatibility with deep learning applications.

The development of our algorithms and simulation environment was executed within the PyCharm integrated development environment. PyCharm's suite of tools and features for Python development allowed for streamlined coding, debugging, and version control management, essential for maintaining the integrity of complex codebases such as those required for deep reinforcement learning simulations.

In terms of hardware, our experiments leveraged an Intel(R) Pentium(R) G3260 @3.30 GHz CPU. Although not the most powerful in the market, this CPU provided sufficient computational capabilities for our simulation needs. The GPU model utilized was the NVIDIA GTX 3090, a high-end graphics card offering substantial parallel processing power, crucial for accelerating neural network computations and graphical simulations.

Our experiments derived their environments from the literature [33], [34]. As illustrated in Fig. 3, we subdivided the environment into meshes, simulating rectangular impediments of various dimensions using eight blocks. Within this setup, the square denotes the initial position or the agent's starting point, and the circle represents the pursuit objective or the target destination. The intended outcome is to determine the most direct route from the starting to the endpoint while circumventing all obstructions. The action refers to the agent's movement decision at any given point within the meshed environment, contingent upon the algorithm's current policy.

B. PARAMETER CONFIGURATIONS

Within the simulation, each grid's side length was standardized at a unitary value, while the diagonal of the grid was set

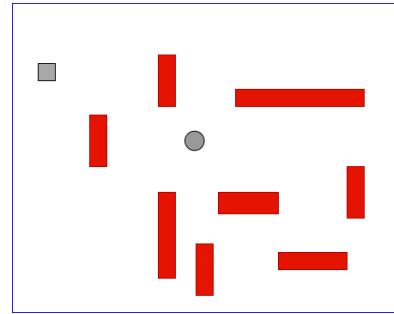


FIGURE 3. Depiction of the simulated environment.

at 20.5 units. The parameter d denotes the constraint distance and is calculated based on the target position identified during preliminary training sessions.

The primary objective of path planning is to ascertain the route that yields the maximum reward upon reaching the goal. To distinguish this from alternate scenarios, a reward value of 50 is allocated when the target is successfully reached. Conversely, colliding with an obstacle incurs a penalty, resulting in a reward value of -5 . In the event the agent is still in the exploration phase, a neutral reward of 0 is accorded when the target remains undetected. Once the target is identified, an appropriate reward is attributed as detailed in [35]. The rewards are categorized into specific scenarios encountered by the agent: achieving the goal, encountering an obstacle, or various states relative to target detection. Specifically, a reward value of 50 is assigned when the agent successfully reaches the goal, signifying the completion of the objective. Conversely, a penalty of -5 is imposed when the agent collides with an obstacle, indicating an undesirable outcome that should be avoided. Additionally, the reward formula r for target-related interactions varies conditionally: it is set to 0 when no target is found, encouraging ongoing exploration, and a cubic function $-0.21 \times (x - 3.12)^3$ is applied when a target is detected but not hit, promoting precision in navigation by adjusting the reward based on the distance x from the target, with 3.12 being a critical distance threshold. This dynamic reward system is designed to optimize the agent's learning process by enhancing its decision-making in complex environments.

In the context of a multilayer perceptron, overfitting may manifest when there's an excessive number of network parameters. Conversely, underfitting might be encountered when these parameters are scanty. For optimal network training, the parameter count is set at 15.

C. PERFORMANCE EVALUATION

Experiments were conducted simulating real-world scenarios. In this environment, the target network parameters were updated at intervals of every 200 iterations. Given the experience pool's initial state devoid of any records, the first 400 training iterations emphasized continual exploration, aiming to populate the pool. Subsequent to these initial sessions, the network engaged in training, randomly drawing

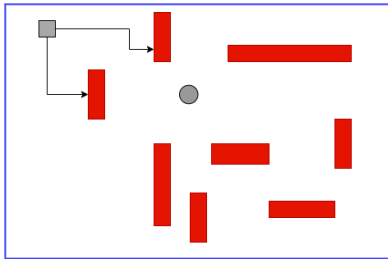


FIGURE 4. Preliminary path finding.

records from the experience pool, and updating its parameters employing the gradient descent method. The initial paths often presented collisions with obstacles, necessitating adjustments towards optimization, as depicted in Fig 4.

As training progressed, augmenting the experience pool, network parameters evolved and the variable ϵ in the ϵ -greedy algorithm incrementally increased by 0.00001 per path exploration, capping at 0.9999. This ensured a gradual convergence from a purely exploratory action selection to a more exploitative approach, optimizing path selection. The culmination of this progression can be observed in Fig 5, illustrating the ideal path.

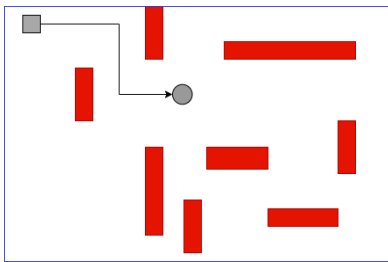


FIGURE 5. Deduced optimal path.

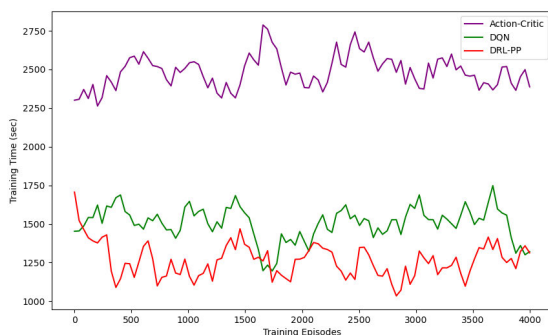


FIGURE 6. Comparison of the average training time.

First, we analyze the training duration, as depicted in the Fig. 6. It offers a comparative overview of different deep reinforcement learning algorithms applied to the domain of path planning.

The Actor-Critic method, represented by the purple line, manifests the most prolonged training duration coupled with pronounced variability, suggesting a suboptimal algorithmic efficiency, particularly in the context of large-scale

applications such as crowd path planning. This inefficiency may stem from the algorithm's inability to adeptly navigate the dynamically shifting policies inherent to multi-agent systems—a challenge compounded by the algorithm's fluctuating policy adjustments.

In contrast, the DQN approach, denoted by the red line, demonstrates a marked reduction in training time relative to the Actor-Critic method, albeit with persisting fluctuations. The enduring oscillations in training time may indicate that the DQN algorithm, while more efficient, still encounters limitations due to each critic network processing solely its respective state-action information. Such a constraint may inadvertently prolong the algorithm's training duration.

The DRL-PP methodology, illustrated by the green line, exhibits the shortest training times and a diminution in fluctuations, underscoring a heightened efficiency and a more stable training progression. This enhanced performance may be attributable to the algorithm's integrated approach, combining centralized learning with decentralized execution within a multi-agent framework. Moreover, the incorporation of a mean field network likely facilitates a reduction in training complexity, optimizing the performance of collaborative planning tasks throughout the training epoch.

In synthesizing the data, it becomes evident that the DRL-PP method surpasses its counterparts concerning both average and minimal training times. Such efficiency is potentially due to the algorithm's sophisticated design, which synergizes centralized learning, decentralized execution, and mean field theory to optimize the training process for complex, multi-agent scenarios. Therefore, DRL-PP stands as the superior path planning strategy within the evaluated DRL methodologies, evidenced by its consistently lower and more stable training times.

Next, we present a juxtaposition of the nascent DRL-PP against the well-established DQN and Actor-Critic frameworks. The comparative discourse pivots on the scrutiny of reward trajectories synthesized by each algorithm across an extensive span of 2,000 training rounds.

Illustrated meticulously in Fig. 7 is the reward trajectory for DRL-PP. Herein, the inception of the algorithm is marked by stochastic explorations, deliberately untethered from experience pool-influenced parameter modifications, rendering the initial reward metrics notably volatile. However, a notable inflection point is observed post the training session mark. The aggregation of experiential insights begins to substantiate parameter updates with enhanced consistency. This progression, in conjunction with a progressively ascending ϵ value, incrementally biases action selection in favor of those yielding maximal rewards, culminating in the emergence of stabilized reward metrics.

Contrastingly, Figs. 8 and 9 illustrate the reward dynamics as characterized by the DQN and Actor-Critic methodologies. These strategies, well-established within the domain of reinforcement learning, exhibit reward trends that, although structurally analogous, show significant divergence in their patterns and stabilization points. The trajectories reveal that

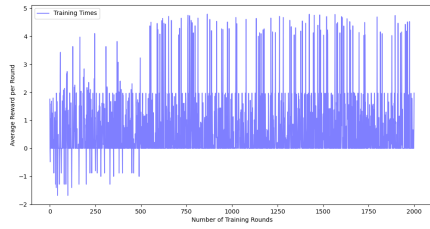


FIGURE 7. DRL-PP algorithm's reward trend over 2,000 iterations.

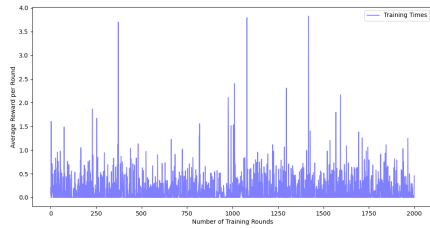


FIGURE 8. Reward trend for the DQN algorithm across 2,000 iterations.

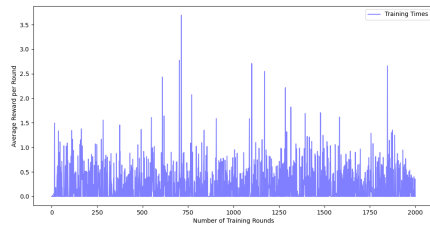


FIGURE 9. Reward trend for the actor-critic algorithm over 2,000 iterations.

while both the DQN and Actor-Critic approaches maintain fidelity to the conventional frameworks of reinforcement learning, the DRL-PP model introduces an evolution towards an advanced learning regime, finely attuned to the complex and dynamic nuances of environment interaction and decision-making processes.

The detailed analysis supports the proposition that DRL-PP, through its synergistic blend of exploratory stochasticity and experience-informed refinement, achieves superior adaptability. This adaptability transcends mere algorithmic robustness, encompassing operational efficiency within real-world applications, thereby highlighting its potential as a pivotal innovation in the field of path planning.

Furthermore, in evaluating performance metrics, particularly for path planning algorithms, efficiency is often measured by the number of exploration planning iterations required to achieve an optimal or satisfactory trajectory. Thus, the count of these iterations serves as an essential metric for assessing the comparative effectiveness of different algorithms.

Within the framework of this paper, both the DQN and Actor-Critic methods were employed as benchmark algorithms to delineate the relative performance of the DRL-PP approach. A consolidated view of the cumulative planning steps taken by each algorithm during their respective

exploration phases is meticulously documented. The data, as encapsulated in Table 2, suggests a distinct advantage in favor of DRL-PP.

TABLE 2. Aggregate planning iterations for each algorithm.

Algorithm	Number of Iterations
Actor-Critic	6700
DQN	5600
DRL-PP	3000

A critical analysis of the tabulated iterations underscores the salient efficiency of DRL-PP, which required significantly fewer iterations to achieve comparable or superior path planning outcomes. This pronounced reduction in exploration iterations not only indicates an enhanced efficiency but also suggests an algorithmic robustness of DRL-PP in navigating complex environments and circumventing obstacles. Consequently, DRL-PP exhibits an elevated efficacy, positioning it as a method of choice for path planning within intricate and dynamically changing landscapes.

Moreover, the lesser number of iterations signifies a lower computational overhead, which in turn translates to faster response times and a more rapid convergence rate. This attribute is particularly valuable in real-time applications where timely decision-making is paramount, such as autonomous vehicle navigation or robotic motion planning in unpredictable terrains. Hence, the efficacy of DRL-PP extends beyond mere iteration counts, impacting the broader spectrum of performance metrics that are vital for real-world deployment of path planning algorithms.

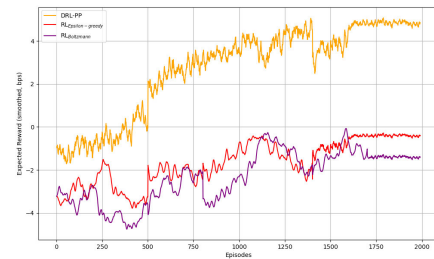


FIGURE 10. Accumulated reward comparison.

Figure 10 provides a systematic depiction of the comparative analysis on Accumulated Reward between a canonical Q-learning construct incorporating ϵ -greedy and Boltzmann exploration tactics, and the DRL-PP methodology. In the initial phase, particularly within the first 500 rounds, the divergence in cumulative rewards among the three algorithms is minimal, indicating a stage where substantive learning has yet to occur. This phase predominantly reflects the impact of initial random explorations, a common characteristic across all tested algorithms, underscoring a realistic assessment of early learning dynamics.

It is pivotal to recognize that within the Q-learning paradigm, the exploration parameters— ϵ for the ϵ -greedy strategy and τ for the Boltzmann strategy—undergo

methodical annealing concurrent with learning progress. This strategic modulation of parameters facilitates a calibrated trade-off between the exploratory and exploitative actions, fundamentally influencing the trajectory and caliber of learning.

The illustrative renditions within the figure succinctly convey the learning pathways synthesized by the Q-learning framework juxtaposed with those of the DRL-PP. The reward trajectory profiles elucidated therein demonstrate that the DRL-PP algorithm exhibits a notably expedited convergence rate relative to the ϵ -greedy Q-learning construct. Even when benchmarked against the more advanced Boltzmann Q-learning approach, the DRL-PP maintains a superior gradient of convergence, reinforcing the robustness and effectiveness of its learning algorithm.

Such empirical evidence distinctively positions the DRL-PP solution as a superior contender in the domain of deep reinforcement learning, showcasing its adeptness in swiftly navigating the complex landscape of strategic decision-making processes.

VI. CONCLUSION

Path planning algorithms are pivotal for mission-oriented decision-making. This significance has catalyzed a surge of interest among researchers, leading to a diverse array of competing methodologies. Given the limitations of traditional single-strategy reinforcement learning networks, which often struggle to efficiently navigate the action space in intricate autonomous path planning scenarios, this paper introduces an enhanced deep reinforcement learning algorithm tailored for path planning in autonomous driving, termed DRL-PP. Empirical results, benchmarked against established DQN-based and Actor-Critic-based path planning algorithms, affirm the superior efficiency of DRL-PP. Notably, it requires fewer exploration steps and discerns the optimal path more expediently within identical environments. The efficiency and efficacy of DRL-PP, as highlighted by these results, signal its substantial potential to advance the capabilities of path planning significantly.

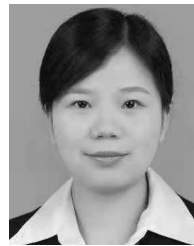
REFERENCES

- [1] C. Katrakazas, M. Qaddus, W.-H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transp. Res. C, Emerg. Technol.*, vol. 60, pp. 416–442, Nov. 2015.
- [2] Y. Ma, W. Liang, J. Li, X. Jia, and S. Guo, "Mobility-aware and delay-sensitive service provisioning in mobile edge-cloud networks," *IEEE Trans. Mobile Comput.*, vol. 21, no. 1, pp. 196–210, Jan. 2022.
- [3] J. Li, Y. Chen, X. Zhao, and J. Huang, "An improved DQN path planning algorithm," *J. Supercomput.*, vol. 78, no. 1, pp. 616–639, Jan. 2022.
- [4] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2092–2104, Feb. 2020.
- [5] Q. Xie, X. Zhang, I. Rekik, X. Chen, N. Mao, D. Shen, and F. Zhao, "Constructing high-order functional connectivity network based on central moment features for diagnosis of autism spectrum disorder," *PeerJ*, vol. 9, Jul. 2021, Art. no. e11692.
- [6] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.
- [7] F. Zhao, Z. Han, D. Cheng, N. Mao, X. Chen, Y. Li, D. Fan, and P. Liu, "Hierarchical synchronization estimation of low- and high-order functional connectivity based on sub-network division for the diagnosis of autism spectrum disorder," *Frontiers Neurosci.*, vol. 15, p. 1898, Feb. 2022.
- [8] Y. Lu and Y. Wan, "Clustering by sorting potential values (CSPV): A novel potential-based clustering method," *Pattern Recognit.*, vol. 45, no. 9, pp. 3512–3522, Sep. 2012.
- [9] G. Gao and R. Jin, "An end-to-end flow control method based on DQN," in *Proc. Int. Conf. Big Data, Inf. Comput. Netw. (BDICN)*, Jan. 2022, pp. 504–507.
- [10] X. Zhang, F. Yang, Y. Guo, H. Yu, Z. Wang, and Q. Zhang, "Adaptive differential privacy mechanism based on entropy theory for preserving deep neural networks," *Mathematics*, vol. 11, no. 2, p. 330, Jan. 2023.
- [11] H. Yu, L. T. Yang, Q. Zhang, D. Armstrong, and M. J. Deen, "Convolutional neural networks for medical image analysis: State-of-the-art, comparisons, improvement and perspectives," *Neurocomputing*, vol. 444, pp. 92–110, Jul. 2021.
- [12] Y. Luo, Y. Zhang, X. Ding, X. Cai, C. Song, and X. Yuan, "StrDip: A fast data stream clustering algorithm using the dip test of unimodality," in *Web Information Systems Engineering—WISE 2018 (Lecture Notes in Computer Science)*, vol. 11234, H. Hacid, W. Cellary, H. Wang, H. Y. Paik, and R. Zhou, Eds. Cham, Switzerland: Springer, 2018, doi: [10.1007/978-3-030-02925-8_14](https://doi.org/10.1007/978-3-030-02925-8_14).
- [13] L. M. Zamstein, A. A. Arroyo, E. M. Schwartz, S. Keen, B. Sutton, and G. Gandhi, "Koolio: Path planning using reinforcement learning on a real robot platform," in *Proc. 19th Florida Conf. Recent Adv. Robot. (FCRAR)*, 2006, pp. 25–26.
- [14] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.
- [15] G. Pan, Y. Xiang, X. Wang, Z. Yu, and X. Zhou, "Research on path planning algorithm of mobile robot based on reinforcement learning," *Soft Comput.*, vol. 26, no. 18, pp. 8961–8970, Sep. 2022.
- [16] A. I. Panov, K. S. Yakovlev, and R. Suvorov, "Grid path planning with deep reinforcement learning: Preliminary results," *Procedia Comput. Sci.*, vol. 123, pp. 347–353, 2018.
- [17] X. Lei, Z. Zhang, and P. Dong, "Dynamic path planning of unknown environment based on deep reinforcement learning," *J. Robot.*, vol. 2018, pp. 1–10, Sep. 2018.
- [18] A. Balachandran, A. S. Lal, and P. Sreedharan, "Autonomous navigation of an AMR using deep reinforcement learning in a warehouse environment," in *Proc. IEEE 2nd Mysore Sub Sect. Int. Conf. (MysuruCon)*, Oct. 2022, pp. 1–5.
- [19] J. Gao, W. Ye, J. Guo, and Z. Li, "Deep reinforcement learning for indoor mobile robot path planning," *Sensors*, vol. 20, no. 19, p. 5493, Sep. 2020.
- [20] A. Konar, I. Goswami Chakraborty, S. J. Singh, L. C. Jain, and A. K. Nagar, "A deterministic improved Q-learning for path planning of a mobile robot," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 5, pp. 1141–1153, Sep. 2013.
- [21] P. K. Das, H. S. Behera, and B. K. Panigrahi, "Intelligent-based multi-robot path planning inspired by improved classical Q-learning and improved particle swarm optimization with perturbed velocity," *Eng. Sci. Technol., Int. J.*, vol. 19, no. 1, pp. 651–669, Mar. 2016.
- [22] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proc. AAAI Conf. Artif. Intell.*, 2016, vol. 30, no. 1, pp. 2094–2100.
- [23] Y. Li, A. H. Aghvami, and D. Dong, "Path planning for cellular-connected UAV: A DRL solution with quantum-inspired experience replay," *IEEE Trans. Wireless Commun.*, vol. 21, no. 10, pp. 7897–7912, Oct. 2022.
- [24] Y. Li, A. H. Aghvami, and D. Dong, "Intelligent trajectory planning in UAV-mounted wireless networks: A quantum-inspired reinforcement learning perspective," *IEEE Wireless Commun. Lett.*, vol. 10, no. 9, pp. 1994–1998, Sep. 2021.
- [25] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.
- [26] H. Yu, L. T. Yang, X. Fan, and Q. Zhang, "A deep residual computation model for heterogeneous data learning in smart Internet of Things," *Appl. Soft Comput.*, vol. 107, Aug. 2021, Art. no. 107361.

- [27] H. Yu, Q. Zhang, and L. T. Yang, "An edge-cloud-aided private high-order fuzzy C-means clustering algorithm in smart healthcare," in *Proc. IEEE/ACM Trans. Comput. Biol. Bioinf.*, 2023, pp. 1–10.
- [28] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, May 1992.
- [29] S. Wang, S. Wang, Z. Liu, and Q. Zhang, "A role distinguishing BERT model for medical dialogue system in sustainable smart city," *Sustain. Energy Technol. Assessments*, vol. 55, Feb. 2023, Art. no. 102896.
- [30] X. Hu, X. Ding, D. Bai, and Q. Zhang, "A compressed model-agnostic meta-learning model based on pruning for disease diagnosis," *J. Circuits, Syst. Comput.*, vol. 32, no. 2, Jan. 2023, Art. no. 2350022.
- [31] X. Libin and C. Chunjie, "A short signal backoff MAC protocol based on game theory for underwater sensor networks," *IEEE Access*, vol. 10, pp. 125992–126000, 2022.
- [32] R. Zhu, Q. Jiang, X. Huang, D. Li, and Q. Yang, "A reinforcement-learning-based opportunistic routing protocol for energy-efficient and void-avoided UASNs," *IEEE Sensors J.*, vol. 22, no. 13, pp. 13589–13601, Jul. 2022.
- [33] F. Cheng, G. Gui, N. Zhao, Y. Chen, J. Tang, and H. Sari, "UAV-relaying-assisted secure transmission with caching," *IEEE Trans. Commun.*, vol. 67, no. 5, pp. 3140–3153, May 2019.
- [34] M. Aljehani and M. Inoue, "Performance evaluation of multi-UAV system in post-disaster application: Validated by HITL simulator," *IEEE Access*, vol. 7, pp. 64386–64400, 2019.
- [35] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," 2016, *arXiv:1611.09940*.



KAI YANG received the bachelor's and master's degrees in computer science from Hohai University, Nanjing, China, in 2004 and 2013, respectively. He is currently an Associate Professor with Nanjing Normal University of Special Education. His research interests include cloud computing, multi-core systems, real-time systems, real-time processing/scheduling, network-on-chip, distributed systems, and the Internet of Things.



LI LIU received the bachelor's degree from Xiangtan University, Xiangtan, China, in 2015, and the master's degree from China Pharmaceutical University, Nanjing, China, in 2018. She is currently a Research Assistant. Her research interests include medical image processing, machine learning, and data mining.

...