Academic Science Journal

Autonomous Agent Learning for Navigation in Continuous Environments: A Deep Reinforcement Learning Approach

Ina'am Rabah Mohammed*[1], Maha Adham Al-Bayati[2] and Farah qais Al khalidi[2]

[1]Al-Sharia Department , College of Islamic Sciences, Diyala University, Diyala, Iraq

[2]Computer Science Department, College of science, Al-Mustansiriya University, Baghdad, Iraq.

m.anaamrabah@uodiyala.edu.iq

Received: 12 March 2024 **Accepted: 1 May 2024**

Published: July 2025

DOI: https://dx.doi.org/10.24237/ASJ.03.03.870A

Abstract

Recently, there has been an increasing interest in applying techniques of machine learning to autonomous agent learning. The autonomous ability of agents to detect and adapt to their environment enhances their adaptability and efficacy in completing numerous activities. A learning framework known as Reinforcement learning (RL) learns an agent how to act in a way that maximizes the value of a reward signal. Within the domain of agentcis, agents are trained to perform various tasks through trial and error, by employing reinforcement learning. In this paper, we utilizing one of the reinforcement learning algorithms, the Deep Q-learning Network (DQN). In DQN, used for autonomous agent learning, the agent can learn and determine the optimum policy that will guide it to its destination from its interactions with the environment. The agent's decision-making ability allows it to modify its policy as a response to observable circumstances. Experiments performed in a simulated continuous environment revealed results that presented the ability of an agent to react in response to changing conditions and demonstrated that the agent was effectively adapting its behavior in accordance with the learned policy.

Keywords: Reinforcement learning, Simulated environment, DQN, Autonomous Learning.

Introduction

The capability of an agent to learn how to navigate and intelligently avoid obstacles represents an essential component of an agent's intelligence. This capability influenced agent movements' efficiency, obstacle information processing, and obstacle avoidance strategies [1]. Traditionally, autonomous agents were designed for model-based environments with clearly defined locations. However, independent agents have to operate in unexpected and unfamiliar situations. One of the most significant features of an agent is the ability to navigate around such environments while avoiding obstacles [2].

One of the critical components of the navigation process is self-navigating autonomous agents. Allowing autonomous agents to learn how to adapt to unexpected and dynamic environments and avoid obstacles in their environment may exhibit intelligent navigation, human-like behaviors, and real-time obstacle avoidance; this guarantees that autonomous agents can securely and successfully operate in challenging real-world systems, such as delivery, transportation, search and rescue, and inspection [3]. The most well-known RL algorithm which was introduced in 1989, is Q-learning [4]. For autonomous agent navigation and obstacle avoidance [5, 6], Q-learning has been applied very successfully by several researchers. The ability of Q-learning to solve path-planning problems for autonomous agents in 3D settings has also been evaluated [7].

Additionally, a number of writers have suggested hybrid techniques that combine fuzzy logic [8, 9] and artificial neural networks [10, 11] with reinforcement learning.

Gao et al. (2020) proposed a reinforcement learning based- agent route planning method. The algorithm first discretizes the information about the direction of target points obtained by LiDAR and the information about obstacles surrounding the autonomous agent into finite states. In this method, the authors designed a continuous reward function and a reasonable number of environment models and state spaces to ensure that the agent's actions are correctly rewarded which increases the training efficiency of the algorithm. The training of an agent carried out in a Gazebo simulation environment, and the effectiveness of the algorithm is confirmed by the training results. Zhang et al. (2020) used the artificial potential field method in combination with reinforcement learning to guide the autonomous agent to complete the path planning task.

This strategy aims to assign a potential value to each point in the agent's path inside the environment. The target point attracts and rewards the agent, while the obstacle repels and penalizes the agent. The main drawback of this method is that it requires a significant amount of a priori knowledge, but it also tends to fall into the problem of local optimality.

Learning-based approaches, which provide considerable promise, try to solve issues with autonomous agent navigation by learning from data. Drivable routes [14], obstacle detectors [15], end-to-end driving from demos [16], [17], and other skills may be learned via supervised learning approaches. On the other hand, the quantity of human-labeled data naturally restricts these techniques, as the capabilities and performance of deep neural networks are typically heavily restricted by the available data. Due to the growing popularity of high-performance computers, reinforcement learning—which learns autonomously via trial and error—achieves remarkable success in a range of activities.

The contribution of this paper presents a deep reinforcement learning system to address the problem of autonomous navigation. By leveraging DQN, autonomous agents can learn navigation policies, make decisions based on sensory inputs, avoid obstacles, follow paths, and reach target locations. The combination of deep neural networks, experience replay, and reinforcement learning techniques in DQN has proven to be effective for autonomous navigation in a wide range of applications, including autonomous vehicles, drones, and mobile robots.

The rest of this paper is arranged as follows: Section 2 presents the algorithm of reinforcement learning used in this paper; Section 3 presents the methodology of the proposed method section 4 highlights the experimental results and discussions. Section 5 is the conclusion.

Reinforcement Learning

Reinforcement Learning (RL) is a powerful learning approach that allows an agent to learn optimal actions by interacting with an environment and receiving feedback in the form of rewards. Deep Q-learning Network (DQN), one of the reinforcement learning methods has shown great success in solving complex tasks, including navigation.

The DQN algorithm approximates the Q-values, which represent the expected cumulative rewards for performing specific actions in different states by utilizing the deep neural network.

DQN can handle high-dimensional state spaces, such as raw sensor inputs like images or point clouds, and learn complex representations directly from the data [18].

One of reinforcement learning's main advantages. Reinforcement learning tackles the larger issue of goal-directed agent-environment interaction, in contrast to some other machine learning techniques that concentrate on discrete sub-problems or particular tasks. It takes into account the full process of an agent learning to make choices with specific objectives in an unpredictable environment. For example, supervised learning concentrates on learning from labeled instances without making it clear how the information acquired would be helpful in reaching a certain objective [19]. In contrast, reinforcement learning highlights the ability of an agent to sense its state, interact with it, and choose behaviors that influence it in order to achieve its goals [20].

Furthermore, some planning techniques take long-term objectives and plan creation into account, although frequently without regard to immediate decision-making. They might not cover the process of acquiring or updating the predictive models required for planning. On the other hand, reinforcement learning combines planning with real-time decision-making By learning the agent to interact with the environment and progressively enhance its decision-making abilities. While those methods have produced a lot of helpful outcomes, a major drawback is that they only address specific sub-problems [21].

This comprehensive approach to reinforcement learning enables agents or any autonomous system to learn and develop strategies for solving complex problems in dynamic and uncertain environments. It makes it possible to integrate observation, action, and decision-making into a coherent framework, which promotes more intelligent and goal-directed behavior [2].

A. Markov decision process

The Markov decision process is a mathematical model of the problem of learning from interaction to achieve an objective. It is the agent who learns and makes decisions. As a result of the agent's actions, the environment responds and shows the agent new situations. Rewards are also produced by the environment and are unique numeric values that the agent aims to optimize progressively by making decisions about what to do [2]. Figure 1 shows a Markov decision process model of the interaction of an agent with an environment.

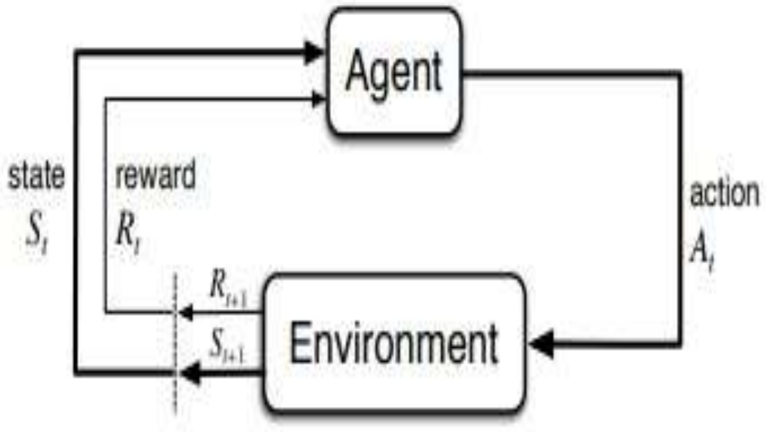

Figure 1: The Interaction of Agent–Environment in a Markov Decision Process Model [2].

B. Q- learning algorithm

A model-free reinforcement learning method for managing Markov Decision Processes (MDPs) is known as Q-learning. An agent is trained using this type of Temporal Difference (TD) learning approach to determine the optimal policy of action for maximizing cumulative rewards in a particular environment.

The expected cumulative rewards that an agent can obtain by carrying out a certain action in a particular state and then following the optimal policy of actions are known as the optimum action-value function Q(s, a), or Q-function, where s denotes the state and a the action [22].

The state-action space is first discovered by the agent through its initial exploration of the environment conducted at random. As training progresses, it exploits the learned Q-values to select actions with higher rewards. The update rule for the Q-value of the current state-action pair is as follows:

$$Q(s, a) = Q(s, a) + \alpha * (r + \gamma * \max[Q(s', a')] - Q(s, a)) \quad (1)$$

where Q(s, a) is the Q-value of state s and action a, α is the learning rate (step size), r is the immediate reward, γ is the discount factor (determines the importance of future rewards), s' is the next state, and max[Q(s', a')] is the maximum Q-value of the next state [23]. The agent can utilize the learned Q-values to determine actions indiscriminately when the Q-values have converged or the training is finished. The optimum policy that maximizes predicted cumulative rewards is obtained by selecting the action with the highest Q-value for each state.

While Q-learning is a popular and effective reinforcement learning algorithm, it also has some limitations and disadvantages.

1. Exploration-Exploitation Tradeoff: Q-learning requires a balance between exploitation and exploration. Initially, for the purpose of choosing the best action, the agent explores its surroundings, yet too much exploration might impede convergence and render learning ineffective. Conversely, premature exploitation resulting from inadequate information may result in inadequate policy. It might be difficult to decide on the best exploration technique, particularly in complicated areas.
2. Curse of Dimensionality: Q-learning struggles when state and action spaces are high-dimensional. Since the dimensionality of the issue increases exponentially with the number of state-action pairs, it is not computationally viable to store and update Q-values for every potential combination. This limitation, also referred to as the "curse of dimensionality," restricts Q-learning's use to situations with small or controllable state-action spaces [24].
3. Lack of Generalization: Q-learning usually learns Q-values for certain pairs of states and actions. It could be difficult to extrapolate from training to previously unforeseen situations or behaviors. This inability to generalize might make it more difficult for the agent to perform well in unfamiliar circumstances or apply what it has learned to similar activities [25].

These drawbacks draw attention to a few of Q-learning's shortcomings and difficulties. It is important to consider that there are Q-learning extensions and variants, including Double Q-learning and Deep Q-Networks (DQN), that solve some of these drawbacks and enhance the algorithm's performance in intricate and high-dimensional situations [26].

C. Deep Q - learning networks (DQN) algorithm

Due to the exponential growth of Q-table's size with the increase in the number of states and actions, one of the primary disadvantages of Q-learning is that it becomes impractical when dealing with enormous state spaces. Deep learning, an important machine learning discipline, was established. Deep learning employs multilayer neural networks to automatically learn the image and extract its detailed features, in contrast to classical machine learning [27]. Combining Q-learning with deep neural networks is an alternate strategy to address this problem. It was introduced by DeepMind in 2015 and has achieved significant success in solving complex reinforcement learning problems [28].

The fundamental concept of DQN is to approximate the action-value function (Q-function) by utilizing a deep neural network, also known as the Q-network. Traditional techniques with high-dimensional and continuous state spaces, and Q-learning struggle; DQN, in contrast, handles these types of situations. The neural network receives the state as an input and outputs the Q-values for all possible actions. The following Figure 2 illustrates the difference between Q-learning and deep Q-network in evaluating the Q-value [29]:

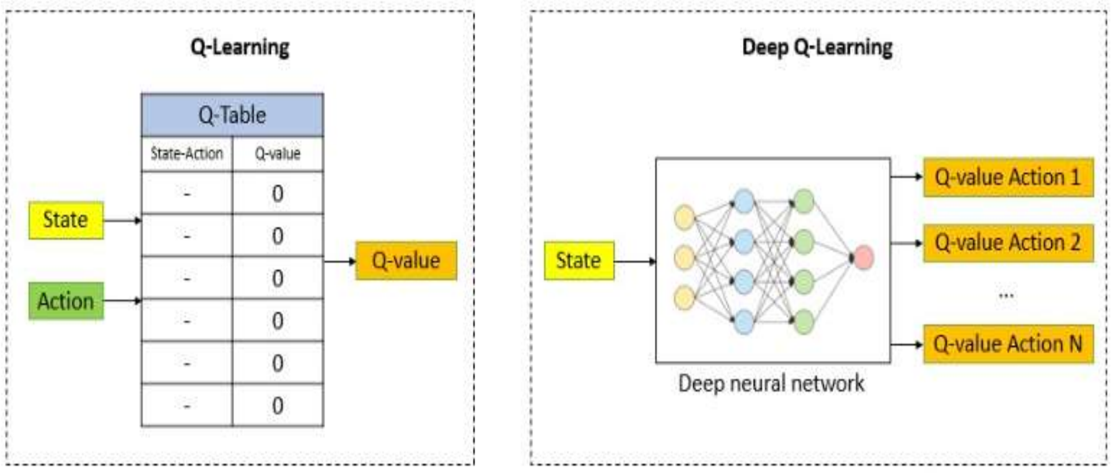

Figure 2: The difference between Q-learning and deep Q-Network [29]

The DQN algorithm introduced several key components to make Deep Q-Learning more stable and efficient, including experience replay and a separate target network.

1. Experience Replay: Developed by DQN, this technique includes randomly sampling mini-batches of experiences during training and storing the agent's experiences in a replay memory buffer to provide more robust learning [29].

2. Target Network: DQN employs not only the Q-network but also an additional target network. While the Q-network is used to choose actions, the target network, which is a replica of the Q-

network, is used to calculate the target Q-values during training. The learning process is stabilized by regularly updating the target network parameters to match the Q-network characteristics. Effective learning requires striking a balance between exploitation—taking the most well-known action—and exploration—trying new actions [28].

Convolutional neural networks (CNNs), often referred to as deep neural networks with convolutional layers [30], are commonly used in DQN as the Q-network [31].

Material and Methods

This section describes the methodology of the proposed method which can be applied in agents, drones or any application in navigation system. The proposed method was tested in a dynamic simulated environment, Figure 3 demonstrates the implementation of our proposed model for autonomous agent learning.

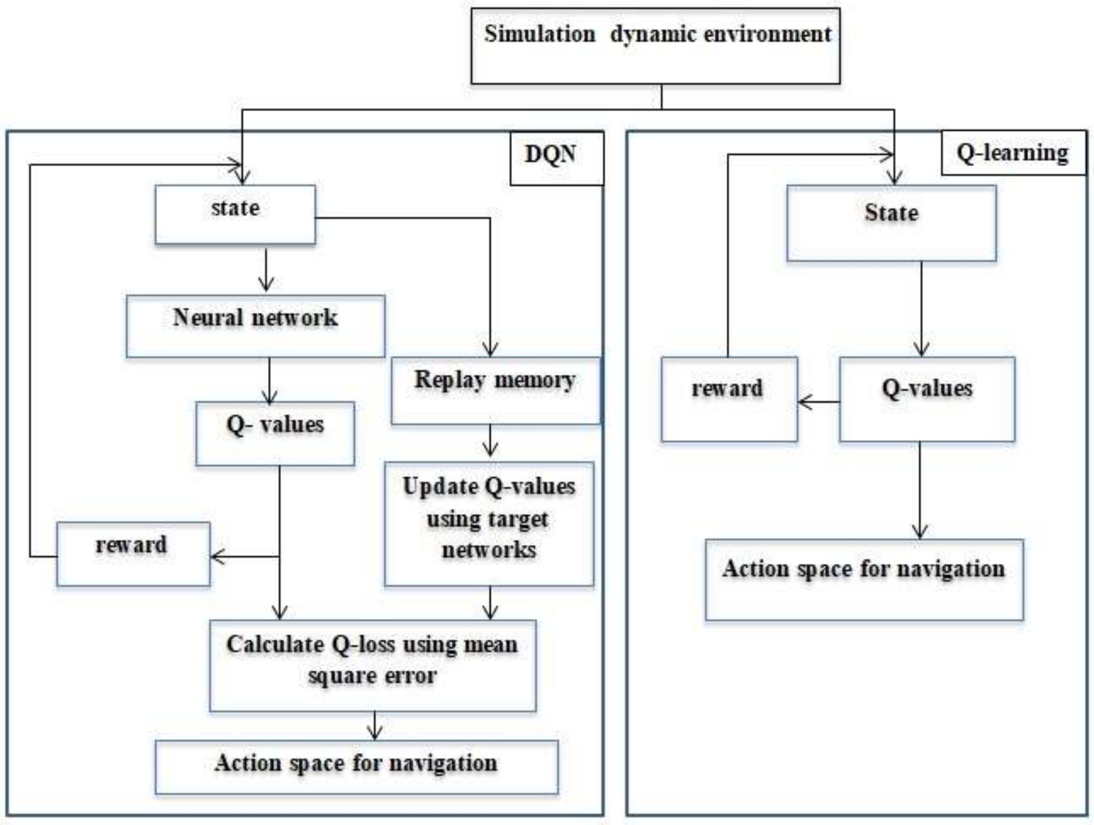
Simulation dynamic environment
DQN
state
Neural network
Replay memory
Q- values
Update Q-values using target networks
reward
Calculate Q-loss using mean square error
Action space for navigation
Q-learning
State
reward
Q-values
Action space for navigation

Figure 3: The diagram illustrating the Structure of the Proposed Model

Results and Discussion

1. Simulation environment

We use the CartPole environment from OpenAI. The actions are 0 to push the cart to the left and 1 to push the cart to the right. The continuous state space is an X coordinate for location, the velocity of the cart, the angle of the pole, and the velocity at the tip of the pole. The score, also called averaged reward, is what we give to the agent to know if its action is good or not. Based on that, the agent will try to optimize and pick the right action. A reward of +10 is granted to the agent at each step while the pole is kept upright. The maximum reward an agent can earn in a single episode is 250. Episode (iteration) ends when episode length exceeds 200 steps.

The states, actions, rewards, and further observations that are gathered throughout the agent's interaction with the environment are what make up the training data in the simulated environment.

2. Training

The DQN algorithm is used as mentioned in section 2. The learning rate α is set to 0.001. A discount factor γ of 0.95 is chosen. The experience replay's batch size was fixed at 24. The agent engages with the environment to gather these state-action-reward-next state transitions during training, then stores them in a replay memory buffer. During the training phase, random transitions are then sampled from this replay memory, which serves to reduce the correlation between successive samples and increase the stability of learning. The training data is used to estimate the Q-values, or predicted rewards, for each state-action combination so that the DQN agent, or any other reinforcement learning agent, may learn from it. The agent updates its Q-values and minimizes Q-loss between the target Q- values and predicted Q-values using mean square error. The mean square error (MSE) equation is commonly used in the training process of the DQN algorithm to update the Q-values. The equation measures the discrepancy between the predicted Q-values and the target Q-values, guiding the agent's learning. It is given by:

$$\text{MSE} = \frac{1}{\text{N}} \sum(\text{Q\_target}(s, a) - \text{Q\_predicted}(s, a))^2 \qquad (2)$$

Where N is the batch size (the number of transitions sampled from the replay memory), Q_target (s, a) represents the target Q-value for a given state-action pair (s, a) and Q_predicted (s, a) represents predicted Q-value for the same state-action pair.

The Q-target value is calculated using the Bellman equation and serves as the desired value for the agent to learn. It is defined as:

$$Q\_target(s, a) = R(s, a) + \gamma * \max Q(s', a') \quad (3)$$

where:

R(s, a) is the immediate reward obtained when taking action in state s, γ is the discount factor that determines the trade-off between immediate and future rewards, max Q(s', a') represents the maximum predicted Q-value for the next states' over all possible actions a'. The Q-predicted value is the agent's estimate of the Q-value for a given state-action pair, obtained from the neural network approximation. The neural network takes the state as input and produces Q-values for each possible action. By minimizing the mean square error, the agent adjusts the weights of the neural network to bring the predicted Q-values closer to the target Q-values, thereby improving its performance over time. Table 1 presents the training process for the 200 episodes.

Table 1: The Training Process of DQN for 200 Episodes.

episodes	score	Q- loss
1/200	23.0	0.9137245
2/200	9.0	0.877809
3/200	10.0	0.839088
4/200	23.0	0.751476
5/200	22.0	0.676394
6/200	19.0	0.618038
7/200	29.0	0.5371084
8/200	13.0	0.5057535
8/200	10.0	0.4834445
9/200	13.0	0.455222
10/200	9.0	0.437329
11/200	9.0	0.4201389
12/200	16.0	0.3897078
13/200	14.0	0.3651230
14/200	10.0	0.3490173
15/200	9.0	0.335298
16/200	12.0	0.317311
17/200	10.0	0.3033145
18/200	11.0	0.288485
19/200	14.0	0.2702863
20/200	12.0	0.255786
21/200	10.0	0.244503
22/200	9.0	0.234893
23/200	10.0	0.212486

24/200	8.0	0.2051603
25/200	9.0	0.1970961
26/200	14.0	0.18466228
27/200	10.0	0.1765167
28/200	12.0	0.15509201
29/200	14.0	0.1466356
30/200	11.0	0.139466
31/200	11.0	0.1326482
32/200	10.0	0.126797
33/200	9.0	0.121204
34/200	10.0	0.1164392
35/200	10.0	0.1113034
36/200	11.0	0.1058624
37/200	8.0	0.1022119
38/200	11.0	0.0972148
39/200	9.0	0.0933936
40/200	11.0	0.0897223
41/200	9.0	0.0861958
42/200	10.0	0.0823937
43/200	10.0	0.0787593
44/200	10.0	0.0752852
45/200	9.0	0.07232597
46/200	27.0	0.06348844
47/200	17.0	0.05859542
48/200	10.0	0.05601075
49/200	23.0	0.05016253
50/200	31.0	0.04315921
.	.	.
.	.	.
.	.	.
.	.	.
299/200	210.0	0.0009945703

Discussions

The results from Table 1 which are presented in Figure 4, show that the decrease in the Q-loss indicates that the agent's Q-values are becoming more aligned with the optimal Q-values, which implies that the agent's policy is improving. As the agent continues to update the Q-network, the predicted Q-values become more accurate approximations of the target Q-values. The Q-loss decreases because the Q-network is learning to better estimate the expected cumulative rewards for each state-action pair.

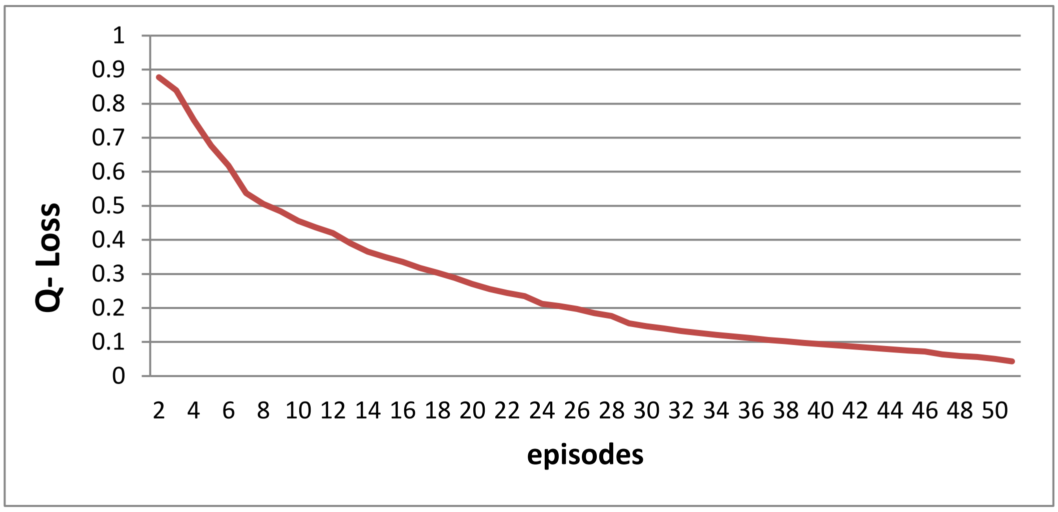

Figure 4: Q-loss per episode during training for 200 episodes

In supervised learning, the performance of a model can be easily tracked during training by evaluating it on the training and validation sets. However, in reinforcement learning, precisely evaluating the progress of an agent during training can be difficult. As suggested by [29], our evaluation metric is the total averaged reward the agent collects in an episode over a number of episodes. The metric of averaged reward tends to be very noisy because small changes in the weights of a policy can lead to large changes in the distribution of states.

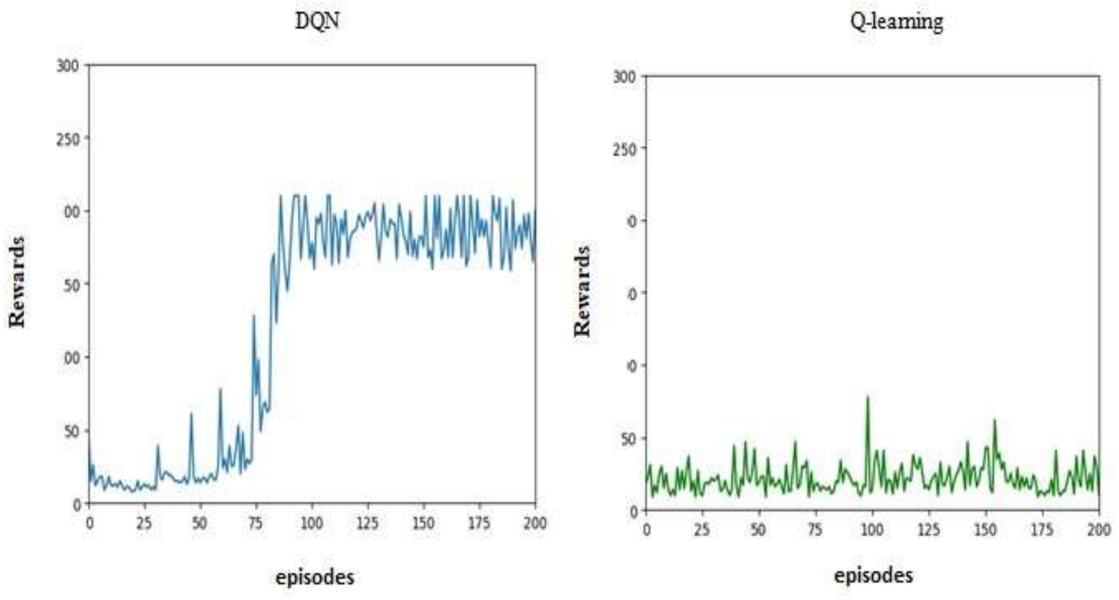

Figure 5: Averaged reward per episode with Q- learning and DQN algorithms

Figure 5 depicts the averaged reward, which can reflect the stability of the DQN algorithm in each iteration. The averaged reward of DQN increases stably while that of Q- learning algorithm varies significantly. The results show that DQN achieves a better balance between exploration and exploitation; this exploration helps the agent discover more rewarding actions and update the Q-network accordingly, leading to a decrease in the Q-loss.

Conclusion

DQN-based autonomous agent learning for navigation in uncertain environments provides a promising approach to train agents to navigate and make decisions adaptively. By combining deep neural networks and Q-learning, agents can learn effective navigation strategies directly from sensor data, leading to more capable and intelligent robotics systems in uncertain and dynamic scenarios. By leveraging DQN-based autonomous agent learning, we can develop intelligent and adaptive robotic systems capable of navigating through uncertain environments with improved efficiency and safety. These systems have the potential to revolutionize various domains, including autonomous vehicles, robotics assistants, and exploration agents.

Source of funding: None

Conflict of interest: None

Ethical clearance: None

References

[1] T. Ribeiro, F. Gonçalves, I. Garcia, G. Lopes, A. F. Ribeiro, Q-learning for autonomous autonomous Mobile robot obstacle avoidance, In: 2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), IEEE, 1-7(2019), DOI(https://doi.org/10.1109/ICARSC.2019.8733621)

[2] R. S. Sutton, and A. G. Barto, Reinforcement learning: An introduction, (MIT press, 201), DOI(https://doi.org/10.1017/S0263574799271172)

[3] F. Gul, W. Rahiman, S. S. N. Alhady, A comprehensive study for agent navigation techniques, Cogent Engineering, 6(1), 1-25(2019), DOI(http://dx.doi.org/10.1080/23311916.2019.1632046)

[4] W. Wang, Z. Wu, H. Luo, and B. Zhang, Path planning method of autonomous robot using improved deep reinforcement learning, Journal of Electrical and Computer Engineering, 2022, 1-7(2022), DOI(https://doi.org/10.1155/2022/5433988)

[5] W. D. Smart, L. P. Kaelbling, Practical reinforcement learning in continuous spaces, In: Proceedings of the seventeenth international conference on machine learning, 903-910(2000)

[6] W. D. Smart, L. P. Kaelbling, Effective reinforcement learning for autonomous robots, In: Proceedings 2002 IEEE International Conference on Robotics and Automation, 4, 3404-3410(2002), DOI(https://doi.org/10.1109/ROBOT.2002.1014237)

[7] Z. Liu, Q. Wang, and B. Yang, Reinforcement Learning-Based Path Planning Algorithm for Autonomous Robots, Wireless Communications and Autonomous Computing, 2022, (2022), DOI(https://doi.org/10.1155/2022/1859020)

[8] H. Boem, and H. Cho, A sensor-based navigation for an autonomous agent using fuzzy logic and reinforcement learning, IEEE Transaction on System, Man, and Cybernetics, 25, 464-477(1995)

[9] N. Yung, and C. Ye, Self-learning fuzzy navigation of autonomous vehicle, In: Proceedings of Third International Conference on Signal Processing, 2, 1465-1468(1996), DOI(https://doi.org/10.1109/ICSIGP.1996.571139)

[10] G. Yang, E. Chen, and C. An, Autonomous agent navigation using neural Q learning, In: IEEE proceedings of the third international conference on machine learning and cybernetics, (2004), DOI(https://doi.org/10.1109/ICMLC.2004.1380601)

[11] K. Macek, I. Petrovic, and N. Peric, A reinforcement learning approach to obstacle avoidance of autonomous agents, In: 7th international workshop on advanced motion control, (2002), DOI(https://doi.org/10.1109/AMC.2002.1026964)

[12] J. Gao, W. Ye, J. Guo, Z. Li, Deep reinforcement learning for indoor autonomous robot path planning, Sensors, 20(19), 5493(2020), DOI(https://doi.org/10.3390/s20195493)

[13] Z. H. Zhengwan, Z. H. Chunjiong, L. I. Hongbing, and X. I. Tao, Multipath transmission selection algorithm based on immune connectivity model, Journal of Computer Applications, 40(12), 3571(2020), DOI(https://doi.org/10.4236/jcc.2024.127007)

[14] D. Barnes, W. Maddern, and I. Posner, Find your own way: Weakly-supervised segmentation of path proposals for urban autonomy, In: 2017 IEEE International Conference on Robotics and Automation (ICRA), 203-210(2017), DOI(https://doi.org/10.1109/ICRA.2017.7989025)

[15] R. Hadsell, P. Sermanet, J. Ben, A. Erkan, M. Scoffier, K. Kavukcuoglu, U. Muller and Y. LeCun, Learning long-range vision for autonomous off-road driving, Journal of Field Agentics, 26(2), 120–144(2009), DOI(https://doi.org/10.1002/rob.20276)

[16] D. A. Pomerleau, Alvinn: An autonomous land vehicle in a neural network, Advances in neural information processing systems, 1, 305–313(1989)

[17] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, and J. Zhang, End to end learning for self-driving cars, arXiv preprint arXiv:1604.07316(2016), DOI(https://doi.org/10.48550/arXiv.1604.07316)

[18] F. Al-Mahamid, and K. Grolinger, Reinforcement Learning Algorithms: An Overview and Classification, In: 2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), 1-7(2021), DOI(https://doi.org/10.1109/CCECE53047.2021.9569056)

[19] A. S. Shaker, S. R. Ahmed, Information Retrieval for Cancer Cell Detection Based on Advanced Machine Learning Techniques, Al-Mustansiriyah Journal of Science, 33(3), 20-26(2022), DOI(https://doi.org/10.23851/mjs.v33i3.1069)

[20] G. Butler, A. Gantchev, P. Grogono, Object-oriented design of the subsumption architecture, Software: Practice and Experience, 31(9), 911-923(2001), DOI(https://doi.org/10.1002/spe.396)

[21] K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, A Survey of Path Planning Algorithms for Autonomous Agents, Vehicles, 3(3), 448-468(2021), DOI(https://doi.org/10.3390/vehicles3030027)

[22] J. Cui, G. Nie, Motion Route Planning and Obstacle Avoidance Method for Autonomous Robot Based on Deep Learning, Journal of Electrical and Computer Engineering, 2022, 1-11(2022), DOI(http://dx.doi.org/10.1155/2022/5739765)

[23] H. Y. Ryu, J. S. Kwon, J. H. Lim, A. H. Kim, S. J. Baek, and J. W. Kim, Development of an Autonomous Driving Smart Wheelchair for the Physically Weak, Applied Sciences, 12(1), 1-18(2021), DOI(https://doi.org/10.3390/app12010377)

[24] J. Raajan, P. V. Srihari, J. P. Satya, B. Bhikkaji, and R. Pasumarthy," Real time path planning of agent using deep reinforcement learning, IFAC-PaperOnLine, 53(2), 15602-15607(2020), DOI(http://dx.doi.org/10.1016/j.ifacol.2020.12.2494)

[25] V. A. Bakale, V. S. YK, V. C. Roodagi, Y. N. Kulkarni, M. S. Patil, S. Chickerur, Indoor Navigation with Deep Reinforcement Learning, In :2020 International Conference on Inventive Computation Technologies (ICICT), 660-665(2020), DOI(https://doi.org/10.1109/ICICT48043.2020.9112385)

[26] R. Singh, J. Ren, and X. Lin, A Review of Deep Reinforcement Learning Algorithms for Autonomous Agent Path Planning, Vehicles, 5(4), 1423-1451(2023), DOI(https://doi.org/10.3390/vehicles5040078)

[27] H. H. Ali, J. R. Naif, and W. R. Humood, A New Smart Home Intruder Detection System Based on Deep Learning, Al-Mustansiriyah Journal of Science, 34(2), 60-69(2023), DOI(https://doi.org/10.23851/mjs.v34i2.1267)

[28] X. Lei, Z. Zhang, and P. Dong, Dynamic path planning of unknown environment based on deep reinforcement learning, Journal of Robotics, 2018(12), 1-10(2018), DOI(https://doi.org/10.1155/2018/5781591)

[29] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller Playing atari with deep reinforcement learning, arXiv:1312.5602v1 [cs.LG], (2013)

[30] N. M. Khassaf, and S. H. Shaker, Image Retrieval based Convolutional Neural Network, Al-Mustansiriyah Journal of Science, 31(4), 43- 54(2020), DOI(http://doi.org/10.23851/mjs.v31i4.897)

[31] K. Yeom, Deep Reinforcement Learning Based Autonomous Driving with Collision Free for Autonomous Agents, International Journal of Mechanical Engineering and Robotics Research, 11(5), 338-44(2022), DOI(10.18178/ijmerr.11.5.338-344)