INCREASED EFFECTIVE OF CONTROL ROBOT USING REINFORCEMENT COMBINE WITH DEEP LEARNING

Luong Thi Thao Hieu*, Pham Thi Thuy, Nguyen Khac Hiep
University of Economic and Technical Industries

ARTICLE INFO

Received: 13/4/2023
Revised: 24/5/2023
Published: 24/5/2023

KEYWORDS

Reinforcement learning
Deep reinforcement learning
Robotic manipulation control
DQN
VGG16

ABSTRACT

Although deep learning can solve problems that cannot be done by tradition machine learning algorithms, it requires huge amount of data, which is not always available in control problems. Reinforcement learning is a good solution in robot control. In reinforcement learning, the data is generated when the agent interacts with environment. Along with the develop of noron networks, many researcher have focused on combine noron network with reinforcement learning to create deep reinforcement learning. In this paper, we propose a new deep reinforcement learning model based on the improvement of the traditional Deep Q Learning algorithm by combining techniques: Fixed_Q Target, Double Deep Q, Prioritized Experience Replay with CNN network (VGG16), apply to control the robot with state and agents designed by using Unity ML-Agents. The proposed model was tested, compared to the original model on the simulation environment. The results show that the proposed method fix the overestimation q value and fast convergence.

NÂNG CAO HIỆU QUẢ ĐIỀU KHIỂN ROBOT, SỬ DỤNG HỌC TĂNG CƯỜNG KẾT HỢP HỌC SÂU

Lương Thị Thảo Hiếu*, Phạm Thị Thùy, Nguyễn Khắc Hiệp
Trường Đại học Kinh tế Kỹ thuật Công nghiệp

THÔNG TIN BÀI BÁO

Ngày nhận bài: 13/4/2023
Ngày hoàn thiện: 24/5/2023
Ngày đăng: 24/5/2023

TỪ KHÓA

Học tăng cường
Học tăng cường sâu
Điều khiển robot
DQN
VGG16

TÓM TẮT

Mặc dù học sâu có thể giải quyết các bài toán mà các thuật toán học máy cũ không giải quyết được nhưng cần lượng dữ liệu rất lớn và trong thực tế dữ liệu không phải lúc nào cũng có sẵn trong bài toán điều khiển. Học tăng cường là một giải pháp tốt trong bài toán điều khiển robot. Trong học tăng cường, dữ liệu được tạo ra khi tác tử tương tác với môi trường. Cùng với sự ra đời của mạng noron, nhiều nghiên cứu đã tập trung kết hợp mạng noron vào học tăng cường tạo nên học tăng cường sâu. Trong bài báo này chúng tôi đề xuất mô hình học tăng cường sâu mới dựa trên sự cải tiến thuật giải Deep Q Learning truyền thống bằng cách kết hợp các kỹ thuật: Fixed_Q Target, Double Deep Q, Prioritized Experience Replay, với mô hình mạng VGG16, ứng dụng điều khiển robot xếp hàng hóa với không gian trạng thái tự thiết kế sử dụng Unity ML-Agents. Mô hình đề xuất được thực nghiệm, so sánh đánh giá hiệu quả so với mô hình ban đầu. Kết quả cho thấy phương pháp đề xuất hội tụ nhanh và khắc phục được hiện tượng ước lượng quá mức giá trị Q.

DOI: https://doi.org/10.34238/tnu-jst.7733

* Corresponding author. *Email: ltthieu@uneti.edu.vn*

1. Giới thiệu

Trong những năm gần đây nhân loại đã chứng kiến sự hồi sinh của học tăng cường, đặc biệt là kết hợp học tăng cường với học sâu, học tăng cường sâu được áp dụng trong nhiều lĩnh vực như game, điều khiển robot [1], [2]. Bài toán điều khiển robot sử dụng học tăng cường sâu là một bài toán trong lĩnh vực AI và robotics, nhằm tìm cách điều khiển sao cho robot học cách tương tác với môi trường để hoàn thành nhiệm vụ hiệu quả và tự động [3]. Năm 2013 nhóm nghiên cứu DeepMind đề xuất Deep Q Network (DQN) [4], thay vì lưu toàn bộ state (trạng thái) trong bảng Q, đã sử dụng mạng noron để ước lượng Q value (giá trị Q). Sử dụng DQN có hạn chế là không ổn định, do chỉ sử dụng một mạng noron để ước lượng cả Q target (giá trị Q mục tiêu) và Q value, nên chỉ một cập nhật nhỏ các trọng số của mạng noron có thể dẫn đến sự thay đổi chiến lược (policy) -là một hàm số ánh xạ trạng thái sang hành động tương ứng, sự phân phối dữ liệu và mối tương quan giữa Q value và Q target. Ngay sau đó nhóm DeepMind tiếp tục đề xuất kỹ thuật Fixed_Q Target [5], ý tưởng bổ sung 2 mạng noron Q, $\hat{Q}$ tại mỗi epoch, thực hiện sao chép các trọng số của mạng Q sang $\hat{Q}$, điều này cố định Q target trong một khoảng thời gian giúp huấn luyện ổn định, tuy nhiên trong một số môi trường ngẫu nhiên, thuật toán này hoạt động kém hiệu quả do thực hiện ước lượng quá mức (overestimation) Q value.

Năm 2016 H.van Hasselt cùng cộng sự giới thiệu thuật toán Double Deep Q Network (DDQN) [6], phương pháp này giải quyết vấn đề ước lượng quá mức Q value, trong DDQN dùng 2 mạng noron Q_1, Q_2 đơn giản, để tách việc lựa chọn hành động và cập nhật Q value riêng biệt, tuy nhiên trong thuật toán này không cố định Q_1, Q_2 nên mặc dù khắc phục được vấn đề xấp xỉ quá mức nhưng vẫn xảy ra quá trình huấn luyện không ổn định.

Năm 2020 Y.Wang và cộng sự nghiên cứu học tăng cường sâu trong điều khiển hành trình robot [7], cũng khoảng thời gian này nhóm nghiên cứu do Pérez-Gil đứng đầu đã sử dụng một mạng noron tăng cường sâu để tìm kiếm hành động tối ưu cho robot [8].

Trong bài báo này chúng tôi nghiên cứu kiến trúc DQN, một số kỹ thuật cải tiến DQN: Fixed_Q Target, DDQN, Prioritized Experience Replay (PER) [9], sau đó kết hợp DQN cùng một số cải tiến trên để học một chiến lược tối ưu (chiến lược ánh xạ trạng thái sang hành động tối ưu), tín hiệu đầu vào từ ảnh thu được từ camera gắn trên robot, sử dụng mạng tích chập VGG16 [10] để trích xuất đặc tính của ảnh. Áp dụng phương pháp đề xuất vào điều khiển tự động robot vận chuyển hàng hóa, khắc phục được vấn đề ước lượng quá mức và quá trình huấn luyện ổn định, hội tụ nhanh.

2. Phương pháp nghiên cứu

Trong nghiên cứu này, nhóm tác giả nghiên cứu lý thuyết về thuật giải DQN và các kỹ thuật cải tiến DQN, sau đó đề xuất một mô hình học tăng cường sâu mới DFDP dựa trên sự kết hợp các kỹ thuật: Fixed_Q Target, DDQN, PER. Hiệu quả cải tiến được đánh giá bằng thực nghiệm điều khiển tự động xe nâng hàng trong môi trường mô phỏng.

2.1. Kiến trúc Deep Q Network [4]

DQN bản chất là một mạng noron nhiều lớp, sử dụng để ước lượng Q value cho mỗi trạng thái tương ứng, với mỗi trạng thái s, đầu ra là một vector chứa các giá trị hành động $Q(s,a;\theta)$ với θ là các tham số của mạng. Công thức ước lượng Q value theo khai triển Bellman, trong DQN như sau:

$$Q_{target}(s,a) = r(s,a) + \gamma \max\left(Q\left(s_{t+1}, a'\right)\right) \quad (1)$$

Trong đó: $Q_{target}(s,a)$ là giá trị hành động Q ước tính cho trạng thái s và hành động a với mô hình mạng noron có trọng số θ.

s_{t+1}: Trạng thái mới mà môi trường chuyển sang sau khi thực hiện hành động a trong trạng thái s.

$r(s,a)$: Phần thưởng nhận được khi agent từ trạng thái s thực hiện hành động a.

γ : Hệ số chiết khấu (nằm từ 0 đến 1)

$\max\left(Q\left(s_{t+1}, a^{'}\right)\right)$: Là giá trị Q lớn nhất được ước tính cho trạng thái s_{t+1} bằng cách sử dụng mạng noron với trọng số θ', trọng số này cập nhật chậm hơn. Hành động được chọn tại trạng thái s_{t+1} được ước tính sử dụng mạng noron với trọng số θ'. DQN sẽ nhận đầu vào là các ảnh, ảnh này được truyền qua mạng và cho đầu ra là vecto Q value cho mỗi hành động tương ứng trong trạng thái đó. Mục đích cần tìm Q value lớn nhất trong vecto đầu ra để đạt được hành động tốt nhất.

2.2. Một số kỹ thuật cải tiến DQN

2.2.1. Fixed_Q Target [5]

Bản chất của cải tiến DQN là dự đoán Q target sao cho sai số giữa Q target và Q value cực tiểu. Công thức ước lượng Q value trong phương trình Bellman (1) nảy sinh vấn đề đó là sử dụng cùng bộ tham số để ước tính Q value và Q target, và kết quả, tại mỗi bước của quá trình huấn luyện Q value thay đổi nhưng Q target cũng thay đổi theo, dẫn đến sự giao động lớn trong huấn luyện.

Fixed_Q Target đề xuất bởi Mnih et al. với ý tưởng sử dụng thêm một mạng $\hat{Q}$ (mạng đích) với tham số θ, mạng này tương tự như mạng Q (mạng chính). Mạng chính được sử dụng để tạo ra các ước tính Q value cho một trạng thái và hành động, mạng đích được sử dụng để cung cấp giá trị mục tiêu để cập nhật mạng chính. Như vậy dùng hai mạng Q và $\hat{Q}$ với mục đích sao chép các trọng số từ Q lưu sang $\hat{Q}$ tại mỗi bước thời gian *t*, điều này sẽ cố định Q target trong một khoảng thời gian giúp quá trình huấn luyện ổn định hơn.

Công thức ước lượng Q value:

$$Q_{target}(s,a) = r(s,a) + \gamma \max\left(\hat{Q}\left(s_{t+1}, a^{'}\right)\right) \quad (2)$$

Trong đó, $r(s,a)$ là phần thưởng nhận được khi chuyển từ trạng thái *s* sang trạng thái s_{t+1} sau khi thực hiện hành động a, γ là hệ số chiết khấu, $\max\left(\hat{Q}\left(s_{t+1}, a^{'}\right)\right)$ là giá trị Q lớn nhất của trạng thái s_{t+1} và các hành động a' được tính toán bởi mạng đích.

2.2.2. Double Deep Q Network (DDQN) [6]

DDQN là một cải tiến của DQN, giúp cải thiện khả năng học, trong DQN công thức ước tính giá trị Q từ phương trình (1) nảy sinh tình huống: Làm thế nào chắc chắn rằng hành động tốt nhất cho trạng thái kế tiếp là hành động đạt Q value tốt nhất. Kết quả là ngay trong giai đoạn đầu của quá trình huấn luyện không đủ thông tin về hành động tốt nhất cần thực hiện, do đó khi lấy Q value cực đại có thể dẫn đến kết quả dương tính giả (false positve). Nếu các hành động không tối ưu thường xuyên cho ra Q value cao hơn Q value của hành động tốt nhất việc huấn luyện trở nên phức tạp, đây là hiện tượng ước lượng quá mức Q value.

DDQN được đề xuất bởi Hado van Hasselt, phương pháp này giải quyết vấn đề ước lượng quá mức giá trị Q value – vấn đề ước tính quá cao hoặc quá thấp giá trị hành động, trong phương pháp này sử dụng 2 mạng noron Q_1 và Q_2 riêng biệt ước tính giá trị của hành động trong một trạng thái, cụ thể Q_1 sử dụng để lựa chọn hành động tối ưu cho trạng thái hiện tại (hành động có Q value cao nhất), trong khi Q_2 được sử dụng để ước tính giá trị Q tương lai (Q target) của hành động được chọn bởi mạng thứ nhất. Công thức ước lượng Q value:

$$Q_2(s,a) = r(s,a) + \gamma * Q_1\left(s_{t+1}, \arg\max\left(Q_2\left(s_{t+1}, a\right)\right)\right) \quad (3)$$

$$Q_1(s,a) = r(s,a) + \gamma * Q_2\left(s_{t+1}, \arg\max\left(Q_1\left(s_{t+1}, a\right)\right)\right) \quad (4)$$

DDQN giúp giảm hiện tượng ước lượng quá mức Q value, kết quả dẫn đến quá trình huấn luyện ổn định hơn, cải thiện khả năng học.

2.2.3. Prioritized Experience Replay (PER) [9]

Trong thuật toán học tăng cường sâu, việc tái sử dụng kinh nghiệm trong quá trình học tương đối quan trọng. DQN sử dụng bộ đệm (replay buffer) để lưu trữ kinh nghiệm trạng thái, hành động, giá trị, phần thưởng và trạng thái tiếp theo. Khi huấn luyện mạng noron, các mẫu được lấy ngẫu nhiên từ bộ đệm này. Mục đích cần tìm kinh nghiệm để sai số giữa Q target và Q value lớn nhất sẽ có độ ưu tiên cao nhất. PER được đề xuất bởi Tom Schaul vào năm 2016, sử dụng PER tăng hiệu quả của việc tái sử dụng kinh nghiệm trong quá trình học, thay vì lưu trữ các trạng thái theo thứ tự, PER sắp xếp các trạng thái theo mức độ ảnh hưởng của chúng đến việc học và lưu trữ với độ ưu tiên cao hơn, hay nói cách khác chính là thay đổi phân phối lấy mẫu, bằng cách sử dụng một tiêu chí để xác định độ ưu tiên (priority) của mỗi bộ kinh nghiệm, định nghĩa như sau:

$$p_i = \left|Q_{target}(s,a) - Q(s,a) + \varepsilon\right| \quad (5)$$

Để tránh overfitting, Tom Schaul đã đề xuất xác suất chọn độ ưu tiên theo công thức sau:

$$P(i) = \frac{{p_i}^{\alpha}}{\sum_{i=1}^{n} {p_i}^{\alpha}} \quad (6)$$

α là tham số điều chỉnh sự ngẫu nhiên, α = 0 : experience sẽ được chọn ngẫu nhiên, α = 1: chọn experience có priority cao nhất

Kết quả là trong mỗi bước thời gian, sẽ có một lô các mẫu được lấy với xác suất trên, các mẫu này được lưu trên bộ đệm ưu tiên (Priorized Experience Replay), được sử dụng để huấn luyện mạng.

2.3. Kết hợp DDQN, fixed_q target và PER, ứng dụng điều khiển tự động xe nâng hàng

2.3.1. Kết hợp DDQN, Fixed_q target, PER

Khi dùng hai mạng Q_1, Q_2 trong DDQN vẫn còn tồn tại hạn chế: do các trọng số trong Q_1, Q_2 bị cập nhật lại sau mỗi lần training dẫn đến: $\left(Q_{1,target}(s,a), Q_{2,target}(s,a)\right)$ thay đổi theo làm cho việc huấn luyện mô hình không ổn định. Do đó kết hợp với kỹ thuật Fixed_Q Target để cố định $\left(Q_{1,target}(s,a), Q_{2,target}(s,a)\right)$ bằng 2 mạng $\hat{Q}_1, \hat{Q}_2$ trong một khoảng thời gian giúp việc huấn luyện ổn định hơn, đồng thời khắc phục được hiện tượng ước lượng quá mức Q value, nhưng khi đó cần đến 4 mạng noron trong đó 2 mạng cần huấn luyện sẽ làm tăng thời gian hội tụ của mô hình.

Giải pháp đề xuất: tận dụng ưu điểm của cả 2 kĩ thuật Fixed_Q Target và DDQN, và chỉ sử dụng 2 mạng noron. Công thức ước lượng Q value như sau:

$$Q(s,a) = r(s,a) + \gamma \hat{Q}\left(s_{t+1}, \arg\max\left(Q(s_{t+1}, a)\right)\right) \quad (7)$$

Đồng thời kết hợp kĩ thuật PER lấy mẫu từ bộ đệm, giúp quá trình huấn luyện hiệu quả hơn.

Mã giả:

Đầu vào: Ảnh với kích thước 256x512x12.

Đầu ra: Q value cho mỗi hành động.

Khởi tạo bộ nhớ *M* với kích thước *N*

Khởi tạo hàm giá trị hành động Q với các trọng số ngẫu nhiên θ.

Khởi tạo hàm giá trị hành động $\hat{Q}$ target với các tham số ngẫu nhiên $\hat{\theta} = \theta$

Khởi tạo các siêu tham số $\alpha, \beta, \varepsilon, \mu$

For *episode = 1* **to** *episode* lớn nhất **do**

Với quan sát s_1

For *t = 1* **to** *T* **do**

Chọn $a_t = \boldsymbol{softmax}(Q_\theta(s_t, .))$

Thực hiện hành động a_t trong mô phỏng, quan sát trạng thái s_{t+1} và phần thưởng r_{t+1} thu được. Lưu trữ giá trị $(s_t, a_t, r_{t+1}, s_{t+1})$ vào bộ nhớ *M*.

Tại mỗi giao dịch, thực hiện lấy *n* mẫu ngẫu nhiên $(s_t, a_t, r_{t+1}, s_{t+1})$ từ bộ nhớ *M*.

Đặt: $y_i = \begin{cases} r_j \text{ if } episode\ end\ step\ j+1 \\ r_j + \gamma \hat{Q}\left(s_{j+1}, \arg\max\left(Q_\theta\left(s_{t+1}, a\right)\right)\right), else \end{cases}$

Tính độ ưu tiên: $p_j = \left(\left|\hat{Q}_\theta\left(s_j, a_j\right) - Q_\theta\left(s_j, a_j\right)\right| + \varepsilon\right)^\alpha$

Thực hiện cập nhật gradient descent với các tham số θ: $\left(n * \frac{p_j^{\ \alpha}}{\sum_{i=1}^{N} p_i^{\ \alpha}}\right)^{-\beta} * \left(y_j - Q_\phi\left(s_j, a_j\right)\right)^2$

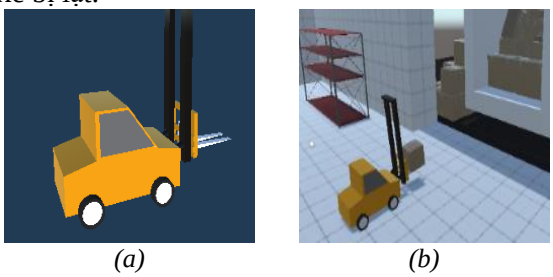
Cập nhật các tham số $\hat{\theta}$ của mạng đích: $\hat{\theta} = (1-\mu) * \hat{\theta} + \mu * \hat{\theta} (0 \le \mu \le 1)$

End
End

2.3.2. Thiết kế không gian mô phỏng

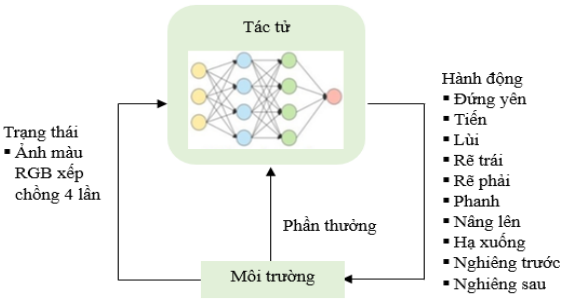
Môi trường mô phỏng được xây dựng bằng game engine Unity và ML-Agent toolkit. Không gian mô phỏng chứa một làn đường, kệ hàng, xe nâng hàng (robot), hàng hóa. Xe nâng hàng 4 bánh, có giá đỡ và thanh ngăn ở phía trước, đồng thời được trang bị camera. Giá đỡ có thể điều chỉnh để đưa hàng lên hoặc xuống, thanh ngăn có thể nghiêng về phía trước hoặc ra sau để tạo độ dốc thuận lợi cho việc di chuyển hàng hoặc đặt hàng lên kệ (hình 1).

Nhiệm vụ của chương trình là phải điều khiển tự động robot đi đến vị trí để nhận hàng và đưa hàng vào các kệ. Mục đích dùng mạng noron sâu kết hợp kỹ thuật học tăng cường, để thu được các giá trị Q value, mỗi giá trị Q value tương ứng với một hành động cần thực hiện của robot. Cụ thể, trình mô phỏng gửi ảnh với 3 kênh màu RGB, xếp chồng 4 lần (thu được từ camera trước của xe nâng hàng) cho tác tử qua localhost. Tác tử lựa chọn action tương ứng, sau đó gửi action đó cho trình mô phỏng. Trình mô phỏng nhận action từ tác tử và thực hiện hành động như cho xe tiến, lùi, rẽ trái, rẽ phải, nghiêng trước (hình 2). Khi robot hoàn thành một nhiệm vụ, agent sẽ nhận được tổng phần thưởng (là một số nguyên) nếu một trong các điều kiện sau xảy ra: +2 robot đi đến đúng chỗ nhận hàng, +5 khi đặt hàng vào đúng kệ, -4 khi đánh rơi hàng trên đường, -5 khi xe bị lật.



(a) *(b)*

Hình 1. *a) Xe nâng hàng, b) Nhà kho*



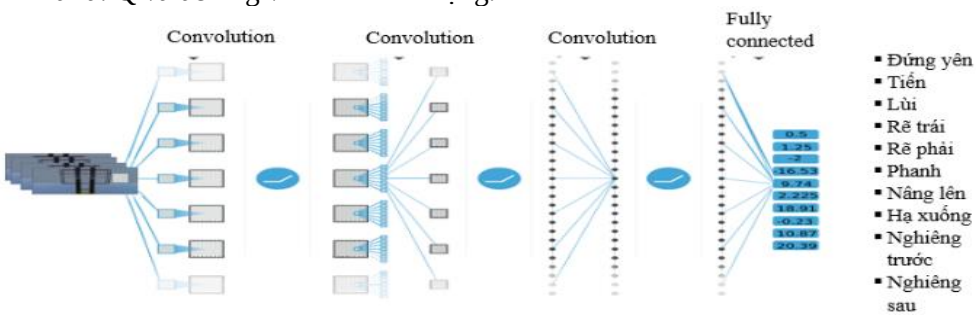
Hình 2. *Mô hình điều khiển robot*

Tại mỗi bước thời gian t, môi trường mô phỏng trả về tensor s_t có số chiều 256x512x12, chương trình điều khiển đưa ra hành động a_t môi trường mô phỏng sẽ di chuyển sang bước thời gian t+1 và trả về phần thưởng r_{t+1} và tensor s_{t+1}. Chương trình lưu $(s_t, a_t, r_{t+1}, s_{t+1})$ ra ổ cứng để huấn luyện.

2.3.3. Kiến trúc mạng điều khiển robot

Kiến trúc mạng điều khiển robot được minh họa trong hình 3, sử dụng mạng tích chập (CNN) được xây dựng dựa trên mô hình VGG16 [10] với 13 lớp tích chập (convolution), số lượng tham số 1.180.778. Ảnh đầu vào được xử lý qua 13 lớp convolution, 5 lớp lấy mẫu maxpooling làm giảm số chiều của ảnh, cuối cùng là 1 lớp kết nối đầy đủ (fully connected) với hàm kích hoạt ELU để chuyển đầu ra từ lớp phía trước thành ma trận có số chiều 10. Trong ma trận này, mỗi giá trị cho ra giá trị ước lượng Q value của mỗi hành động, tổng cộng có 10 hành động: 0. Đứng yên, 1. Tiến, 2. Lùi, 3. Rẽ trái, 4. Rẽ phải, 5 Phanh, 6. Nâng lên, 7. Nâng xuống, 8 Nghiêng trước, 9. Nghiêng sau.

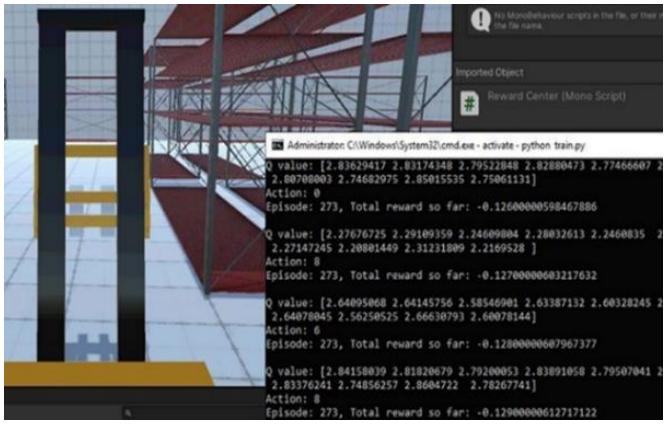
- Đầu vào: Ảnh màu với kích thước 256 x 512 x 12 xếp chồng 4 lần.
- Đầu ra: Q value ứng với mỗi hành động.



Hình 3. *Kiến trúc mạng noron sử dụng điều khiển robot*

Sau khi sử dụng mạng noron sâu cho ra các giá trị Q value, tương ứng với các hành động của robot, sử dụng công thức (7) của kỹ thuật học tăng cường đã đề xuất để ước lượng giá trị Q target, giá trị này sẽ dùng để điều chỉnh Q value.

2.3.4. Tinh chỉnh tham số



Hình 4. *Xe thực hiện hành động nghiêng trước*

Sau khi xây dựng xong mô hình, tiếp theo tinh chỉnh các tham số: hàm kích hoạt, hàm tối ưu, tốc độ học, batch size, epoch. Trong mô hình này sử dụng hàm kích hoạt ReLU cho lớp tích chập, hàm kích hoạt ELU cho lớp fully-connected. Để huấn luyện mô hình sử dụng thuật toán tối ưu adam, triển khai adam với các tham số: $learningrate = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 1e-7$

Thực nghiệm huấn luyện mô hình với 200 episode (mỗi episode sẽ kết thúc khi xe bị lật), mạng noron được huấn luyện với số epoch là 1 và batch size là 32, từ tập dữ liệu gồm 128 mẫu lấy từ bộ đệm theo kỹ thuật PER tại mỗi bước thời gian (timestep).

Trong hình 4, minh họa ảnh chụp một thời điểm chạy thực nghiệm. Kết quả đầu ra là các vecto Q value có số chiều 10, với vecto có giá trị Q value lớn nhất bằng 2,31231809 (ở vị trí thứ 8), khi đó xe thực hiện hành động nghiêng trước.

3. Kết quả và thảo luận

Triển khai phương pháp đề xuất trong mô phỏng điều khiển xe nâng hàng, đánh giá hiệu quả của phương pháp đề xuất (để thuận tiện ký hiệu DFDP) với DQN truyền thống sử dụng độ đo:

+ mean Q value là giá trị trung bình của các giá trị Q (được tính bằng lấy tổng các giá trị Q của tất cả các hành động trong một trạng thái chia cho số lượng hành động), đánh giá mean Q value theo thời gian là cách để theo dõi mức độ học của mô hình, nếu mean Q value tăng theo thời gian đó là tín hiệu cho thấy mô hình đang học được cách ước tính Q value tốt trong các trạng thái khác nhau, ngược lại: Nếu mean Q value giảm theo thời gian cho thấy mô hình đang gặp khó khăn.

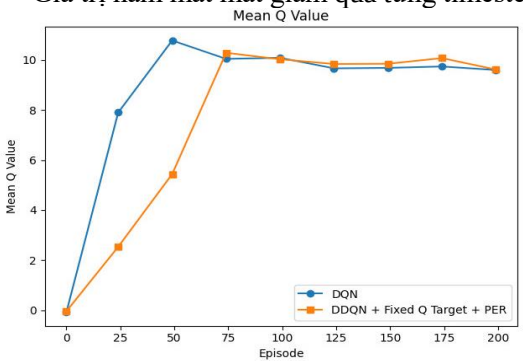
+ total reward: Tổng phần thưởng thu được trong quá trình tương tác với môi trường, đây là thông số quan trọng để đánh giá hiệu quả thực hiện các hành động trong học tăng cường, nếu robot thực hiện hành động hiệu quả (đi đến chỗ nhận hàng, đặt hàng vào kệ), tổng phần thưởng càng lớn.

+ Sử dụng hàm mất mát (loss function) mean squared error (MSE), để đánh giá mức độ hội tụ của mô hình.

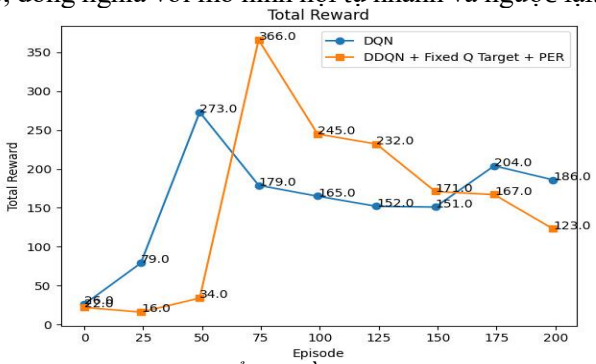
$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 \quad (8)$$

Trong đó n là số lượng điểm dữ liệu, y_i là giá trị đầu ra thực tế của điểm dữ liệu thứ i, $\hat{y}_i$ là giá trị đầu ra dự đoán của điểm dữ liệu thứ i.

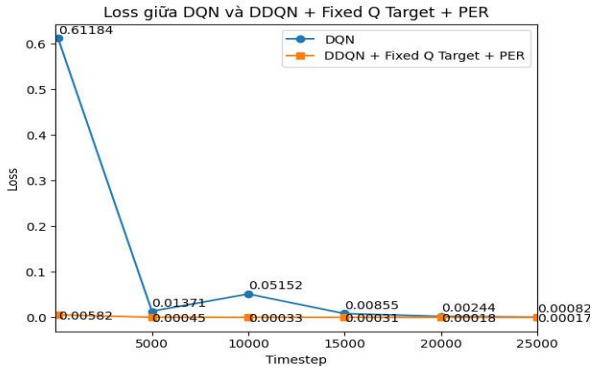
Giá trị hàm mất mát giảm qua từng timestep, đồng nghĩa với mô hình hội tụ nhanh và ngược lại.



Hình 5. *Trung bình giá trị Q của DQN và DFDP*



Hình 6. *Tổng phần thưởng DQN và DFDP*



Hình 7. *So sánh giá trị hàm mất mát giữa DQN và DFDP*

Quan sát hình 5, từ episode 0-50, 2 mô hình cho mean Q value tăng khá nhanh, DQN tăng nhanh hơn so với DFDP, do các trọng số chưa được tối ưu vì vậy các Q value ước tính trong giai

đoạn này không chính xác và có thể khác xa so với Q value thực tế. Khi mô hình được tiếp tục huấn luyện thì cả 2 đều hội tụ tại giá trị 10 trong episode 50 – 75. Từ episode 75-200 mean Q value của DFDP tăng nhẹ dần đều, điều này cho thấy mô hình đang ước tính Q value tốt (không xảy ra hiện tượng xấp xỉ quá mức) đồng nghĩa với quá trình huấn luyện ổn định, ngược lại trong DQN, mean Q value biến động lên xuống và giảm dần (mô hình đang gặp khó khăn trong quá trình huấn luyện.

Quan sát biểu đồ hình 6, từ các episode 75-150 (giai đoạn ổn định), các giá trị tổng phần thưởng của DFDP đều lớn hơn các giá trị tổng phần thưởng của DQN. Từ episode 150-200 (giai đoạn cuối của quá trình huấn luyện) giá trị tổng phần thưởng của DQN lớn hơn DFDP. Trong trường hợp tốt nhất tổng giá trị tổng phần thưởng của DFDP đạt giá trị 366, của DQN là 273. Qua đó cho thấy sử dụng DFDP, robot đã đưa ra các hành động hiệu quả hơn DQN trong môi trường mô phỏng.

Từ biểu đồ hình 7 cho thấy giá trị hàm mất mát đánh giá trên tập huấn luyện của mô hình DFDP giảm dần đều theo các bước thời gian và thấp hơn so với DQN, trường hợp tốt nhất loss của DFDP là 0,00017 của DQN là 0,00082, điều này chứng minh DFDP hội tụ nhanh hơn DQN.

4. Kết luận

Nghiên cứu học tăng cường sâu ứng dụng trong điều khiển là vấn đề phức tạp và thu hút nhiều nghiên cứu. Dựa trên các nghiên cứu về thuật giải học tăng cường sâu DQN, các kỹ thuật cải tiến DQN: Fixed_Q Target, DDQN, PER, chúng tôi đề xuất một mô hình học tăng cường sâu mới DFDP dựa trên sự kết hợp các kỹ thuật: Fixed_Q Target, DDQN, PER, đồng thời sử dụng mô hình học sâu VGG16 để xử lý ảnh đầu vào thay cho CNN đơn giản trong thuật toán gốc. Áp dụng mô hình xây dựng điều khiển xe nâng hàng, thực nghiệm cho thấy mô hình đề xuất khắc phục được hiện tượng ước lượng quá mức giá trị q (quá trình huấn luyện ổn định) và hội tụ nhanh hơn DQN.

TÀI LIỆU THAM KHẢO/ REFERENCES

[1] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement Learning in Robotics: A Survey," *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238-1274, 2013.

[2] R. Liu, F. Nageotte, and P. Zanne, "Deep reinforcement learning for control of robotic manipulation: A review," *Journal of Robotics*, vol. 10, no. 1, pp. 1-14, 2020.

[3] A. Barzegar and D.-J. Lee, "Deep Reinforcement Learning-Based Adaptive Controller for Trajectory Tracking and Altitude Control of an Aerial Robot," *Applied Sciences*, vol. 9, pp. 1-8, 2022.

[4] V. Minh, K. Kavukcuoglu, D. Silver, and A. Graves, "Playing Atari with Deep Reinforcement Learning," *Noron Information Proccessing Systems*, vol. 26, pp. 2674-2682, 2013.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, and A. Graves, "Human level Control through Deep Reinforcement Learning," *Nature*, vol. 518, pp. 529-533, 2015.

[6] H. V. Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-Learning," *in AAAI Conference on Artifical Intelligence (AAAI)*, 2016, pp. 2094-2100.

[7] C. A. W. Y. Wang, "A Deep Reinforcement Learning Method for Robot Dynamic Path Planning," *IEEE Access*, vol. 8, pp. 33817-33826, 2020.

[8] Ó. Pérez-Gil, R. Barea, E. López-Guillén, L. M. Bergasa, C. Gómez-Huélamo, R. Gutiérrez, and A. Díaz-Díaz, "Deep reinforcement learning based control for Autonomous Vehicles in CARLA," *Multimed Tools Appl*, vol. 81, no. 3, pp. 3553–3576, 2022

[9] T. Schaul, "Prioritzed Experience Replay," *in ICLR*, 2016, pp.1-8.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional network for large-scale image recognition," *in The 3rd International Conference on Learning Representations(ICLR2015)*, 2015, pp. 1-14.