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

INCREASED EFFECTIVE OF CONTROL ROBOT USING REINFORCEMENT COMBINE WITH DEEP LEARNING

Lương Thị Thảo Hiếu, Phạm Thị Thùy

Khoa Công nghệ thông tin, Trường Đại học Kinh tế - Kỹ thuật Công nghiệp

Đến Tòa soạn ngày 06/03/2023, chấp nhận đăng ngày 19/04/2023

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, dữ liệu được tạo ra khi tác tử tương tác với môi trường. Cùng với sự ra đời của mạng neural, nhiều nghiên cứu đã tập trung kết hợp mạng neural 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. Thực nghiệm, so sánh đánh giá hiệu quả mô hình đề xuất 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.

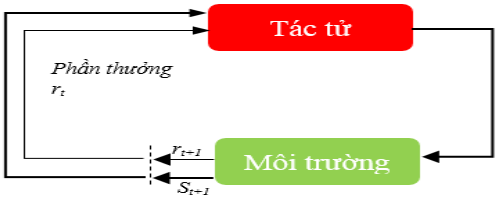
Từ khóa: Học tăng cường, học tăng cường sâu, học sâu, điều khiển robot, DQN, VGG16.

Abstract: Although deep learning can solve problems that cannot be done by tradition machine learning algorithms, it requires huge a amount of data, which is not always available in control problems. Reinforcement learning is a good solution in robot control, the data that is generated when the agent interacts with environment. Along with the develope of neural networks, many reseracher have focused on combine neural 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. Experimentally, comparing and evaluating the effectiveness of the proposed model with the original on the simulation environment, the test results show that the proposed method fix the overestimation q value and fast convergence.

Keywords: Reinforcement learning, Deep reinforcement learning, Robotic Manipulation Control, DQN, VGG16.

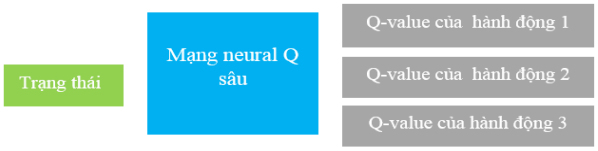
1. GIỚI THIỆU

Học tăng cường là phương pháp học mà các tác tử (agent) tương tác với môi trường (enviroment) học một chính sách (policy) tối ưu thông qua cách thử và sai, và nhận phần thưởng tích lũy theo từng hành động thực hiện. Học tăng cường được sử dụng trong các bài toán ra quyết định tuần tự.



Hình 1. Kịch bản học tăng cường

Các agent cập nhật các tham số (trạng thái - state, hàm giá trị - value function) khi tương tác với môi trường, các tham số này lưu trong bộ nhớ dưới dạng bảng (Q table), điều này kém hiệu quả với môi trường có không gian trạng thái lớn như trong các bài toán điều khiển robot. Ý tưởng tốt nhất trong trường hợp này là với một trạng thái cho trước, sử dụng một mạng neural Q sâu (Deep Q Neural Network) để xấp xỉ các giá trị Q (Q value) - chất lượng của một hành động thực hiện trong một trạng thái nhất định. Mỗi trạng thái được biểu diễn dưới dạng một vecto đầu vào, đầu ra là Q value cho tất cả các hành động có thể thực hiện trong trạng thái đó.



Hình 2. Sơ đồ học tăng cường sâu

Cùng với sự phát triển của công nghệ học sâu, nhân loại đã chứng kiến sự hồi sinh của học tăng cường, đặc biệt là kết hợp học tăng cường với học sâu.

Năm 2013 nhóm nghiên cứu DeepMind đề xuất Deep Q Learning [1] là một cải tiến của Q Learning thay vì lưu toàn bộ state trong Q table đã sử dụng mạng neural để ước lượng Q value. Sử dụng Deep Q Learning có hạn chế là không ổn định, sự không ổn định xuất phát từ mối tương quan trong chuỗi quan sát, do chỉ sử dụng một mạng neural để ước lượng cả Q target (giá trị mục tiêu) và Q value, nên chỉ một cập nhật nhỏ các trọng số của mạng neural có thể dẫn đến sự thay đổi chính sách (policy), phân phối dữ liệu và mối tương quan giữa Q value và Q target. Ngay sau đó nhóm DeepMind tiếp tục đề xuất sử dụng kỹ thuật Fixed_Q Target [2], ý tưởng bổ sung 2 mạng neural Q, $\hat{Q}$ tại mỗi epoch, thực hiện sao chép các trọng số của Q sang $\hat{Q}$, điều này sẽ 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 [4], phương pháp này giải quyết vấn đề ước lượng quá mức Q value, trong thuật toán này 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 tác giả không cố định Q_1, Q_2 nên mặc dù khắc phục được vấn đề overestimation nhưng vẫn xảy ra quá trình huấn luyện không ổn định.

Trong những năm gần đây học tăng cường sâu được nghiên cứu áp dụng trong nhiều lĩnh vực như game, điều khiển robot [9], [12]. Bài toán điều khiển robot sử dụng học tăng cường sâu là một bài toán trong lĩnh vực AI và robotics, mục tiêu của bài toán này là tìm cách điều khiển robot sao cho robot học cách tương tác với môi trường để hoàn thành nhiệm vụ hiệu quả và tự động. Để đạt mục tiêu này, học tăng cường sâu là phương pháp hiệu quả nhất. Năm 2020 Y. Wang và cộng sự nghiên cứu học tăng cường sâu trong điều khiển hành trình robot [10], cũng khoảng thời gian này nhóm nghiên cứu do N.A. Khan đứng đầu đã đề xuất giải pháp điều khiển robot vận chuyển hàng hóa bằng học tăng cường sâu, phương pháp này sử dụng một mạng neural tăng cường sâu để tìm kiếm hành động tối ưu cho

robot giảm thiểu việc overfitting, tăng cường tính tổng quát của mô hình [11].

Trong bài báo này chúng tôi nghiên cứu kiến trúc Deep Q Network (DQN) [1], một số kỹ thuật cải tiến DQN: Fixed_Q Target [2], Double Deep Q Network (DDQN) [4], Prioritized Experience Replay (PER) [5], sau đó kết hợp giữa DQN truyền thống cùng một số cải tiến trên để học thành công các chính sách từ tín hiệu đầu vào từ ảnh thu được từ camera gắn trên robot, sử dụng mạng tích chập VGG16 [3] để trích xuất đặc tính của ảnh. Áp dụng phương pháp đề xuất vào bài toán điều khiển tự động robot vận chuyển hàng hóa, thực nghiệm cho thấy phương pháp đề xuất khắc phục được vấn đề ước lượng quá mức giá trị q (quá trình huấn luyện ổn định), giá trị hàm mất mát giảm dần theo thời gian (hội tụ nhanh).

2. ĐIỀU KHIỂN ROBOT SỬ DỤNG HỌC TĂNG CƯỜNG SÂU [7]

2.1. Bài toán điều khiển robot sử dụng học tăng cường sâu

a) Không gian trạng thái: được mô tả bởi các thông số như vị trí, vận tốc, góc quay

b) Bộ điều khiển: Được lập trình để thực hiện các hành động cần thiết, bản chất là sử dụng mạng neural sâu để xấp xỉ các trạng thái và hành động của robot.

c) Mục tiêu: Robot di chuyển đến vị trí lấy hàng, thực hiện nâng hàng và đưa lên kệ.

Mục đích của học tăng cường sâu trong bài toán điều khiển robot là huấn luyện mạng neural sâu để dự đoán chuỗi các hành động tối ưu, nghĩa là sử dụng mạng neural để ước tính các hàm giá trị trạng thái và hàm giá trị hành động: $\hat{v}(s;\theta)$, $\hat{Q}(s,a;\theta)$, với θ là các trọng số của mạng neural sao cho tổng giá trị phần thưởng (Total reward) trong quá trình tương tác với môi trường là lớn nhất có thể.

2.2. Mô hình động học của xe nâng hàng

2.2.1. Mô hình robot xe nâng hàng

Xe nâng hàng 4 bánh, có giá đỡ và thanh ngăn ở phía trước, đồng thời được trang bị camera. Giá đỡ có thể nâng lên hoặc xuống, thanh ngăn có thể nghiêng về phía trước hoặc ra sau.

(a) (b)

Hình 3. a) Xe nâng hàng b) Nhà kho

Có tổng cộng 18 kệ hàng được đặt xung quanh nhà kho bao gồm 3 ngăn để đặt hàng. Nhiệm vụ của chương trình là phải điều khiển robot tự động đi đến đúng vị trí để nhận hàng và đưa hàng vào các kệ.

2.2.2. Mô hình động học

Trong bài toán điều khiển robot vận chuyển hàng hóa bằng học tăng cường sâu, phần điều khiển chia làm hai phần:

- Sử dụng một số phương trình động lực học thực hiện điều khiển bánh xe di chuyển:

- Để tăng tốc: $\tau_l = d * mF$, $\tau_r = d * mF$

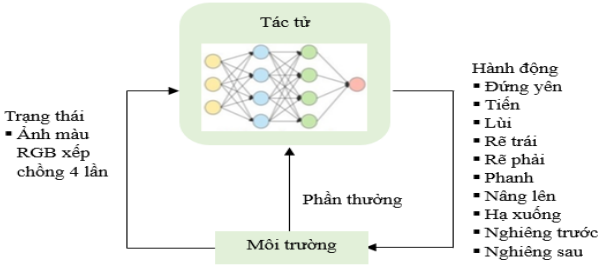
Trong đó τ_l, τ_r là lực momen xoắn tác động lên bánh xe bên trái và bên phải; d: điều chỉnh hướng quay của bánh xe: d = 1, quay bánh xe theo chiều kim đồng hồ, và xe di chuyển tiến, d = −1, quay bánh xe ngược chiều kim đồng hồ, xe di chuyển lùi; mF: Lực quay bánh xe.

- Chỉnh góc lái: $\alpha = \alpha * sd$

Với α: góc lái của bánh xe; sd: Điều chỉnh hướng của góc lái, nếu sd = 1 lấy góc lái ngược chiều kim đồng hồ (rẽ trái), sd = −1 là

lấy góc lái theo chiều kim đồng hồ (rẽ phải).

- Sử dụng mô hình học sâu: Mục đích dùng mạng neural sâu, để thu được các giá trị q value, mỗi giá trị q value sẽ tương ứng với một hành động cần thực hiện của xe nâng hàng. Cụ thể, trình mô phỏng gửi cho tác tử hình ảnh RGB có kích thước 128x256 thu được từ camera trước của xe nâng hàng 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 tương ứng như cho xe tiến, lùi, rẽ trái, rẽ phải, hạ xuống, nghiêng trước…



Hình 4. Mô hình điều khiển robot vận chuyển hàng hóa

Trong hình 4 thông tin đầu vào là trạng thái hiện tại của đối tượng: là một ảnh RGB được chụp từ camera trước có kích thước 128×256.

Đầu ra của mạng là một hành động cho biết các lệnh điều khiển nào sẽ được thực hiện để điều khiển chuyển động, chẳng hạn lệnh nâng lên, hạ xuống, nghiêng trước, nghiêng sau. Khi robot hoàn thành một nhiệm vụ, Agent sẽ nhận được phần thưởng nếu một trong các điều kiện sau xảy ra: +2 robot đi đến đúng chỗ nhận hàng, +5 khi đặt hàng vào đúng kệ hàng, –0.001 tại mỗi timestep, –4 khi đánh rơi hàng trên đường, –5 khi xe bị lật.

3. KIẾN TRÚC DEEP Q NETWORK (DQN) VÀ MỘT SỐ KỸ THUẬT CẢI TIẾN [1] [2] [4] [5]

3.1. Kiến trúc Deep Q Network [1]

DQN bản chất là 1 mạng neural 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, nếu không gian trạng thái có *n* chiều, và không gian hành động *m* chiều, mạng neural là một hàm từ R^n sang R^m. Công thức ước lượng Q value theo khai triển Bellman, trong DQN:

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

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 neural có trọng số θ; s_{t+1}: Là 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: Là hệ số chiết khấu (nằm từ 0 đến 1); $\max\left(Q\left(s_{t+1}, a^{'}\right)\right)$: Giá trị Q lớn nhất được ước tính cho trạng thái s_{t+1} bằng cách sử dụng mạng neural 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 neural 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.

3.2. Một số kỹ thuật cải tiến DQN

3.2.1. Fixed_Q Target [2]

Đã có nhiều chiến lược cải tiến DQN, 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 DQN sử dụng phương trình Bellman (1), tuy nhiên vấn đề nảy sinh đó là sử dụng cùng bộ tham số để ước tính Q value và Q target, kết quả là 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 được 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, trong khi 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 mạng Q lưu sang mạng $\hat{Q}$ 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:

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

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.

3.2.2. Double Deep Q Network (DDQN) [4]

DDQN là một cải tiến của DQN, giúp cải thiện khả năng học. Trong DQN sử dụng một mạng neural để ước tính giá trị Q cho các hành động trong một trạng thái, công thức ước tính giá trị Q từ phương trình Bellman (1) nảy sinh vấn đề sau: 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 neural Q_1 và Q_2 riêng biệt ước tính giá trị của hành động trong một trạng thái, cụ thể mạng Q_1 sử dụng để lựa chọn hành động tối ưu (tốt nhất) cho trạng thái hiện tại (hành động có Q value cao nhất), trong khi mạng 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:

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

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

DDQN giúp giảm hiện tượng ước lượng quá mức Q value, đồng nghĩa với quá trình huấn luyện ổn định hơn, cải thiện khả năng học.

3.2.3. Prioritized Experience Replay (PER) [5]

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 của các trạng thái, hành động, giá trị, phần thưởng và trạng thái tiếp theo. Khi huấn luyện mạng neural, các mẫu được lấy ngẫu nhiên từ bộ đệm này để huấn luyện. Tuy nhiên các mẫu trong bộ đệm không có giá trị như nhau đối với quá trình học, các mẫu tương tác với môi trường trong quá khứ gần đây có thể có giá trị cao hơn đối với việc cập nhật mạng so với các mẫu tương tác trong quá khứ lâu đời hơn, điều này có thể làm giảm hiệu suất học. Mục đích cần tìm kinh nghiệm để giá trị sai số giữa Q target và Q value lớn nhất sẽ có độ ưu tiên cao nhất. Do thực hiện lấy mẫu theo lô (batch) -

lựa chọn các kinh nghiệm một cách ngẫu nhiên - nên những kinh nghiệm quan trọng thực tế không phải lúc nào cũng được lựa chọn. PER được đề xuất bởi Tom Schaul vào năm 2016, ý tưởng một vài kinh nghiệm (experience) quan trọng hơn kinh nghiệm khác nhưng có tần suất xuất hiện ít hơn so với kinh nghiệm bình thường, sử dụng PER tăng hiệu quả của việc tái sử dụng kinh nghiệm trong quá trình học. Giải pháp trong PER, 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 các trạng thái này 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 hiện tượng overfitting, Tom Schaul đã đề xuất xác suất chọn độ ưu tiên theo công thức sau: $P(i) = \frac{p_i^\alpha}{\sum_{i=1}^{n} p_i^\alpha}$ (6)

α là tham số điều chỉnh sự ngẫu nhiên;

$\alpha = 0$: experience sẽ được chọn ngẫu nhiên;

$\alpha = 1$: 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.

4. KẾT HỢP DDQN, FIXED_Q TARGET VÀ PER, ỨNG DỤNG ĐIỀU KHIỂN ROBOT

4.1. 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 neural 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 2 kĩ thuật Fixed_Q Target và DDQN, và chỉ sử dụng 2 mạng neural. 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 thêm kĩ thuật PER để lấy mẫu từ bộ đệm phát lại, 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:

$$p_i = \left(\left|\hat{Q}_\theta\left(s_j, a_j\right) - Q_\theta\left(s_j, a_j\right)\right| + \varepsilon\right)^\alpha$$

Thực hiện cập nhật gradient descent với các tham số θ:

$$\left(y_i - Q_\theta\left(s_j, a_j\right)\right)^2$$

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

4.2. Thiết lập 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 một làn đường, một nhà kho, xe nâng hàng, hàng hóa.

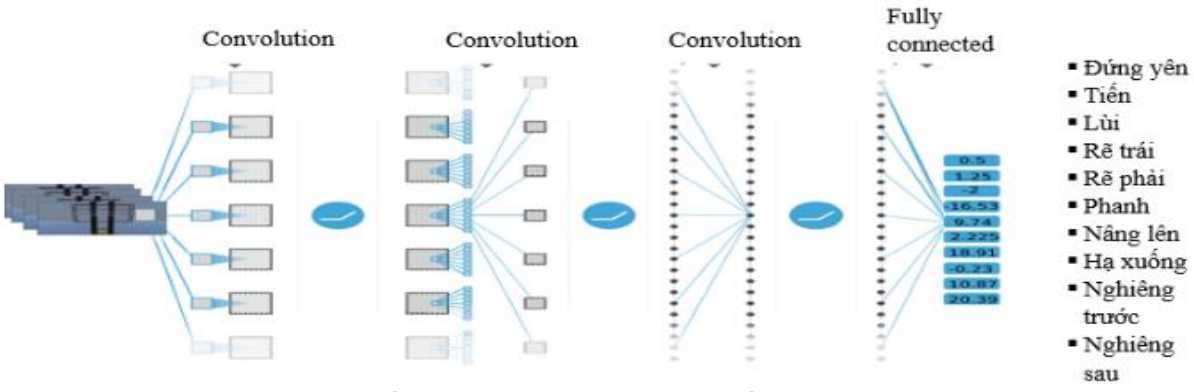
Dữ liệu thu được liên tục từ camera trước của xe nâng hàng là hình ảnh với 3 kênh màu RGB xếp chồng 4 lần, dựa vào hình ảnh đó xe phải đưa ra hành động tương ứng. Tại mỗi bước thời gian t, môi trường mô phỏng trả về tensor s_t có số chiều 256×512×12, chương trình điều khiển đưa ra hành động a_t môi trường 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 kinh nghiệm $(s_t, a_t, r_{t+1}, s_{t+1})$ ra ổ cứng để huấn luyện.

4.3. Kiến trúc mạng điều khiển robot

Kiến trúc mạng được minh họa trong hình 5: Sử dụng mạng CNN được xây dựng dựa trên mô hình VGG16 [3] 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, 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 × 12 xếp chồng 4 lần.
- **Đầu ra**: Q value ứng với mỗi hành động.

Sau khi sử dụng mạng neural 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.



Hình 5. Kiến trúc mạng neural sử dụng điều khiển robot

4.4. Tinh chỉnh siêu tham số

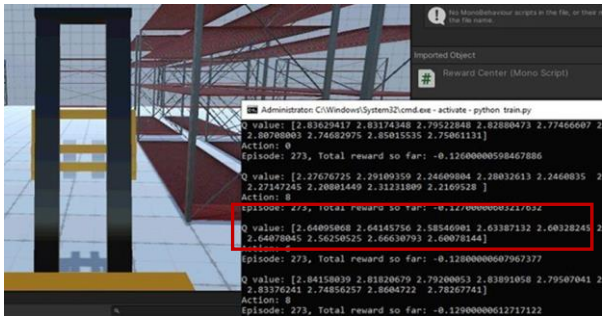
Tinh chỉnh tham số là tìm tập tham số phù hợp nhất để xây dựng mô hình từ tập dữ liệu. Các tham số cần tinh chỉnh là: hàm kích hoạt, hàm tối ưu, tốc độ học, batch size, epoch. Trong mô hình này sử dụng hàm kích hoạt ReLU cho lớp tích chập, và hàm kích hoạt ELU cho lớp fully -connected.

Để huấn luyện mô hình cần sử dụng thuật toán tối ưu, chúng tôi sử dụng thuật toán tối ưu adam [6], 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 neural 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 6 minh họa ảnh chụp một thời điểm thực nghiệm. Kết quả đầu ra là các vecto Q value có số chiều 10, xe nâng hàng căn cứ giá trị lớn nhất trong vecto này để thực hiện hành động và nhận phần thưởng tương ứng



Hình 6. Robot thực hiện hành động nghiêng trước

5. KẾT QUẢ VÀ THẢO LUẬN

Triển khai phương pháp đề xuất trong môi trường mô phỏng điều khiển xe nâng hàng, đánh giá, so sánh 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 một 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) [7], để đánh giá mức độ hội tụ của mô hình.

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

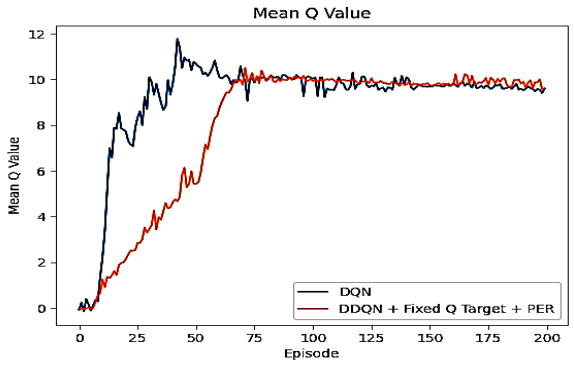
Trong đó n là số lượng điểm dữ liệu, y_i là giá trị đầu ra thực tế của điểm dữ liệu thứ i, $\hat{y}_i$ là giá trị đầu ra dự đoán của điểm dữ liệu thứ i.

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

Quan sát hình 7, từ episode 0-50, 2 mô hình cho mean Q value tăng khá nhanh, trong giai đoạn này 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 mô hình đều hội tụ tại giá trị 10 trong episode 50 – 75.

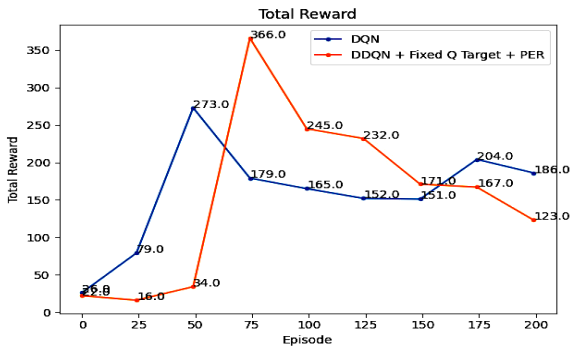
Bảng 1. Biến động trung bình Q value của DQN và DFDP

Thuật giải \ Episode	75	100	125	150	175	200
DQN	10.039	10.082	9.661	9.680	9.736	9.431
DFDP	10.281	10.023	10.0834	9.844	10.069	10.0854



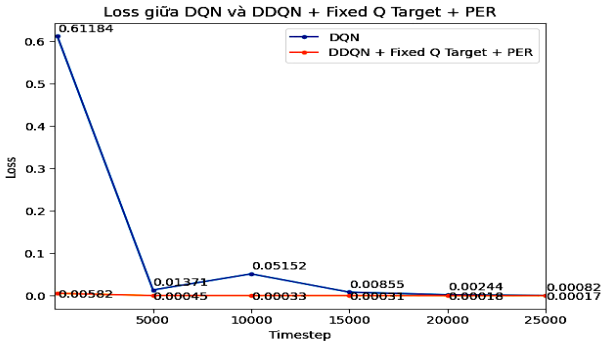
Hình 7. Trung bình giá trị Q của DQN và DFDP

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 nghĩa là mô hình đang gặp khó khăn trong quá trình huấn luyện (bảng 1).



Hình 8. Tổng phần thưởng của DQN và DFDP

Quan sát biểu đồ hình 8, 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 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.



Hình 9. So sánh giá trị hàm mất mát giữa DQN và DFDP

Từ hình 9 cho thấy giá trị hàm mất mát (loss) trên tập huấn luyện của 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.

6. 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 hình học tăng cường sâu mới DFDP dựa trên sự kết hợp: Fixed_Q Target, DDQN, PER, đồng thời sử dụng mô hình học sâu VGG16 xử lý ảnh đầu vào thay cho CNN đơn giản trong thuật toán gốc. Áp dụng mô hình 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 và hội tụ nhanh hơn DQN.

TÀI LIỆU THAM KHẢO

[1] V. Mnih, etal.,*"Playing Atari with Deep Reinforcement Learning"*, in Advanceds in neural information proccessing systems, vol 26, pp. 2674-2682,2013.

[2] V. Mnih, et al., "*Human level Control through Deep Reinforcement Learning*", Nature, vol. 518, pp. 529-533, 2015.

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

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

[5] T. Schaul, "*Prioritzed Experience Replay*", in ICLR, 2016.

[6] Kingma, D.P. & Ba,J. "Adam: A method for stochastic optimazation". In International Conference on Learning Representations (ICLR), Sandiego, CA, USA,2016.

[7] S. Ruder,*"An Overview of Gradient Descent Optimization Algorithm"*, arXvid :1609.04757, 2016.

[8] L. Schmidhuber, et al., "*Reinforcement Learning in Robotics: A Survey*", International Journal of Robotics Research, vol. 36, no. 13-14, pp. 1425-1457, 2017.

[9] R. Liu, "*Deep reinforcement learning for the control of robotic manipulation: A review*", Journal of Robotics, pp. 1-14, 2020.

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

[11] N.A. Khan, A.L.Prodromou and A.Gasteratos,"*Deep reinforcement learning-based robot control for good transportation*", in 2020, 8th International Conference on Control, Mechatronics and Automation (ICCMA), Czech Republic, pp.444-449, 2020.

[12] X. Zhang, W.Wang and X.Yang, "*integrating model predictive control and deep reinforcement learning for robot path planning*," in EEE Transactions on Cybernetics, vol.51, no.2, pp. 973-982, Feb 2021

Thông tin liên hệ: **Lương Thị Thảo Hiếu**

Điện thoại: 0942160880 - Email: ltthieu@uneti.edu.vn

Khoa Công nghệ thông tin, Trường Đại học Kinh tế - Kỹ thuật Công nghiệp.