Multi-Agent Reinforcement Learning for Adversarial-Cooperative Tasks: A DDQN Approach

1st Nguyễn Đức Huy
22022655

2nd Vũ Minh Hiếu
22022610

I. Introduction

Trong những năm gần đây nhiều bài toán khó nhằn như xe tự hành, robot logistic tự hành đã được giải quyết nhờ vào sự phát triển không ngừng của học tăng cường đa tác tử (MARL). Thần kỳ là vậy nhưng học tăng cường hiện chưa thể được coi là hòn đá của nhà giả kim, nó có những khó khăn khi xử lý các ràng buộc phân tán cũng như sự phức tạp của môi trường gia tăng theo cấp số nhân khi số lượng tác tử tăng. Tuy nhiên, nếu có thể vượt qua được những khó khăn này cộng với tiềm năng gần như vô hạn của mình, MARL có thể sẽ là chìa khóa để con người mở ra cánh cửa dẫn đến AGI và xa hơn là ASI.

Trải qua một học kỳ 15 tuần, chúng em sẽ vận dụng những kiến thức đã tiếp thu trong quá trình học tập để giải bài toán **adversarial - cooperative multi-agent reinforcement learning** trong môi trường battle_v4 từ thư viện magent2; cụ thể là sử dụng **Double Deep Q Network** như là phương pháp chính. Phương pháp này sử dụng hai neural network: Q-network và target network, trong đó Q-network là một mạng tích chập nhận đầu vào là quan sát tác tử từ đó cho ra giá trị của từng hành động, target network là mạng có cấu trúc giống hệt Q-network có vai trò ổn định quá trình học bằng cách cập nhật tham số từ tham số của Q-network sau một khoảng thời gian. Kết hợp với việc sử dụng Replay Buffer - là bộ nhớ để lưu trữ các transition của từng tác tử, phục vụ cho quá trình huấn luyện.
Phương pháp này đã đạt hiệu quả tốt chỉ sau 4 episode trong thời lượng huấn luyện khoảng 1h30p. Kết quả evaluate trên 30 episode với max step của mỗi agent là 300:

Bảng I
Kết quả chính

Model	Win	Draw	Lose
Random	30 ± 0	0 ± 0	0 ± 0
Pretrain-0	30 ± 0	0 ± 0	0 ± 0
New Pretrain	30 ± 0	0 ± 0	0 ± 0

Kết quả trung bình thu được cho từng model:

- Random: 30
- Pretrain-0: 30
- New Pretrain: 30

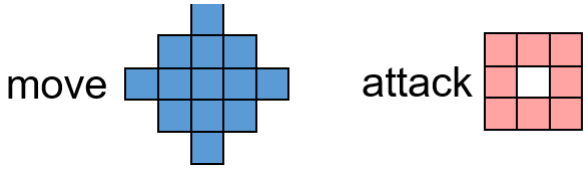
II. Related Work

MARL(Multi-Agent Reinforcement Learning) ngày càng nhận được nhiều sự quan tâm từ cộng đồng các nhà nghiên cứu. Nhiều nhóm nghiên cứu đã và đang tập trung vào việc thiết kế môi trường chất lượng đi kèm với những bài toán có độ phức tạp cao, kết hợp thêm các yếu tố: quan sát một phần(partial observability), thử thách động (challenging dynamics) và quan sát đa chiều (high-dimensional observation).
Môi trường Keepaway soccer (Stone et al., 2005) [3] là một môi trường mô phỏng bóng đá 2D dựa trên RoboCup. Nhiệm vụ chính của tác tử trong môi trường là giữ bóng trong khu vực xác định. Kế thừa Keepaway soccer là môi trường Half Field Offense (Kalyanakrishnan et al., 2006; Hausknecht et al., 2016) [1], độ khó tăng lên khi yêu cầu được đặt ra là ghi bàn. Hai môi trường được đề cập có tính đơn giản, khả năng nâng cấp độ khó hạn chế khi số lượng tác tử gia tăng dẫn đến hai môi trường thường được sử dụng làm testbed cơ bản cho nghiên cứu MARL. Phức tạp hơn đôi chút chúng ta có môi trường gridworld. Lowe et al. (2017) đã giới thiệu một môi trường gridworld cơ bản cho MARL với MADDPG, trọng tâm bài toán nằm ở giao tiếp (shared communication) và điều khiển cấp thấp (low level continuous control). Ngoài ra Resnick et al. (2018) cũng đề xuất một môi trường được xây dựng dựa trên tựa game Bomberman, bao gồm nhiệm vụ hợp tác và đối kháng với độ phức tạo cao trong thế giới gridworld.

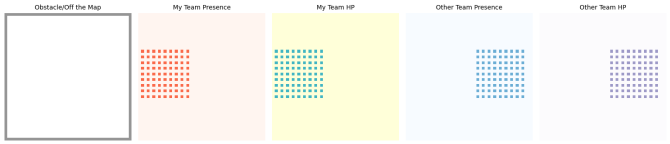
Còn rất nhiều môi trường MARL với độ phức tạp cực cao và sẽ tiêu tốn một lượng lớn tài nguyên tính toán để triển khai, tiêu biểu và nổi bật có TorchCraft (Synnaeve, 2016) [4] và SC2LE (Vinyals, 2017) [5]. Do hạn chế về nhân lực, vật lực cũng như thời gian dành cho dự án, nhóm quyết định sử dụng môi trường battle_v4 từ thư viện magent2 để xây dựng và giải bài toán đối kháng - hợp tác trong môi trường đa tác tử.

III. Methods

Môi trường battle_v4 là một gridworld có không gian trạng thái 45*45*5 trong đó môi kích cỡ môi trường là hình vuông 45*45 và 5 channels tức 5 thuộc tính của môi trường bao gồm: tường, sự hiện diện của tác tử đỏ, máu của tác tử đỏ, sự hiện diện của tác tử xanh, máu của tác tử xanh. Môi trường cũng cung cấp 2 đội xanh, đỏ với 81 tác tử mỗi bên có id từ "red_0"đến "red_80", từ "blue_0"đến "blue_80", mỗi tác tử có 21 hành động rời rạc: di chuyển các hướng và tấn công các hướng (Hình 1, 2). Môi trường hoạt động bằng cách duyệt tuần tự qua từng tác tử của 2 đội và để tác tử đó hành động,

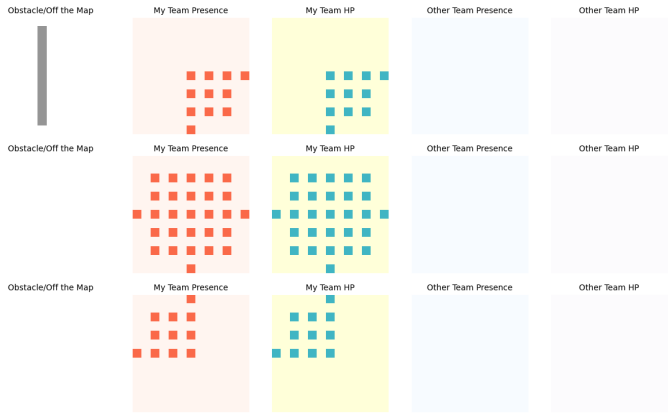


Hình 1. Hành động của mỗi tác tử



Hình 2. Không gian trạng thái

phần thưởng được trao cho hành động cá nhân thay vì cả đội, tác tử sẽ hồi 0.1 máu mỗi lượt, và bị tấn công sẽ khiến tác tử mất 2 máu mỗi lần. Quan sát của mỗi tác tử là quan sát một phần (partial observation) với không gian quan sát là 13*13*5 có thuộc tính tương tự không gian trạng thái (Hình 3). Dựa trên đặc điểm và cách hoạt động của môi trường, dự án này đã thiết kế và cài đặt những nội dung sau.



Hình 3. Không gian quan sát của một số tác tử.

1) **Kiến trúc mô hình**
Đầu tiên là về kiến trúc của mô hình, nhóm sử dụng mô hình neural network với kiến trúc tích chập (CNN) nhằm đảm bảo tác tử có thể xử lý và trích xuất đầy đủ các chi tiết từ quan sát của môi trường (quan sát có kích thước 13*13*5).
Dưới đây là mô tả chi tiết về luồng tiến trình hoạt động của mô hình:
   - Trước hết, hai lớp tích chập đầu tiên đều sử dụng kích thước bộ lọc là 3*3 . Đầu ra của mỗi lớp đều đi qua hàm kích hoạt ReLU để tăng tính phi tuyến và khả năng học các đặc trưng phức tạp.
   - Đầu ra sau đó được duỗi (flatten) để đưa vào lớp fully connected.
   - Lớp fully connected đầu tiên đưa đầu vào đến không gian ẩn với số chiều là 120. Sau đó, lớp fully connected thứ hai thu gọn đầu ra của lớp trước đó về số chiều của hành động. Giữa các lớp này cũng dùng hàm kích hoạt ReLU để duy trì tính phi tuyến.

2) **Phương pháp huấn luyện** Sau khi đã thiết kế xong kiến trúc mô hình, tiếp đến là quá trình huấn luyện. Phương pháp Q-learning được cải tiến bằng cách đưa mạng học sâu vào để ước lượng hàm giá trị (Deep Q-learning). Huấn luyện với tác tử chơi ngẫu nhiên bên đỏ. Quá trình huấn luyện diễn ra theo các bước sau:
   a) **Chiến lược chọn hành động (Epsilon-Greedy Policy)**
   Tác tử chọn hành động dựa trên chiến lược epsilon-greedy như sau:
      - Chọn hành động ngẫu nhiên để khám phá với xác suất ϵ.
      - Chọn hành động tối ưu dựa trên q-values được ước lượng từ mạng chính có xác suất $1 - \epsilon$.

   Giá trị của ϵ giảm dần theo số tập được mô tả bởi công thức:

$$\epsilon = max(\epsilon_{end}, \epsilon_{start} - (\epsilon_{start} - \epsilon_{end})\frac{steps}{\epsilon_{decay}})$$

   Với việc giá trị được điều chỉnh (cụ thể là giảm) theo số tập huấn luyện. Kinh nghiệm thu được từ những hành động, gồm có trạng thái, hành động, phần thưởng và trạng thái tiếp theo, cần phải được lưu trữ. Do đó **Replay buffer** đã được lựa chọn làm giải pháp.
   b) **Replay Buffer:** Kinh nghiệm thu được của từng tác tử gồm quan sát, hành động, phần thưởng, quan sát tiếp theo và trạng thái kết thúc của tác tử đó $(s, a, r, s', done)$ sẽ được lưu trữ trong một bộ nhớ được thiết kế riêng. Cụ thể được thực hiện như sau:
      i) Dữ liệu được lưu trữ theo dạng FIFO đảm bảo kinh nghiệm mới nhất được lưu trữ và kinh nghiệm cũ nhất bị loại bỏ nếu đầy bộ nhớ.
      ii) Lấy mẫu dữ liệu: các lô dữ liệu được lấy ngẫu nhiên từ bộ nhớ phục vụ cho quá trình huấn luyện, việc lấy mẫu là ngẫu nhiên trong bộ nhớ để đảm bảo phân bố độc lập, giống nhau của dữ liệu huấn luyện (i.i.d), tránh việc dữ liệu bị bias, cải thiện hiệu suất huấn luyện.

   Sau khi kinh nghiệm được lưu và lấy mẫu từ Replay Buffer, tiếp đến chúng em cần một hàm loss để huấn luyện mạng.
   c) **Hàm loss**
   Hàm loss được chọn là hàm Huber, sử dụng để tính sai số giữa q-values của mạng chính policy network và mạng mục tiêu (target network):

$$L = \text{SmoothL1Loss}\left(Q(s,a), r + \gamma \max_{a'} Q'(s',a')\right)$$

   Trong đó:
   $Q(s,a)$ là q values hiện tại.
   $Q'(s',a')$ là q values mục tiêu.
   γ là hệ số chiết khấu của phần thưởng tương lai.

r là phần thưởng sau hành động.
a là hành động của tác tử.
Các trọng số thu được của mạng policy network được tính toán dựa trên hàm loss Huber cần trải qua quá trình tối ưu.

d) **Hàm tối ưu**
Nhóm lựa chọn hàm tối ưu AdamW kết hợp với gradient clipping để cập nhật trọng số của mạng chính. Đưa ra sự kết hợp như vậy để tránh gradient quá lớn dẫn đến mất ổn định quá trình huấn luyện. Sau khi đã cập nhật trọng số mạng chính bằng AdamW và gradient clipping, mạng mục tiêu cũng cần được cập nhật. Tiếp theo chúng em sẽ sử dụng phương pháp soft update để cập nhật mạng mục tiêu.

e) **Soft update cho mạng mục tiêu**
Do kiến trúc mô hình, mạng luôn được cập nhật trọng số sau mỗi bước di chuyển của tác tử , do đó nhóm đã sử dụng phương pháp soft update mạng mục tiêu với hệ số τ theo công thức sau:

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta'$$

Trong đó:
θ là trọng số mạng mục tiêu .
θ' là trọng số mạng chính.
τ là hệ số soft update

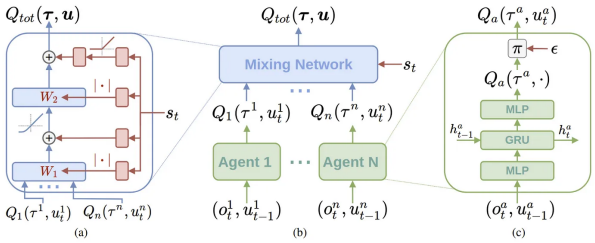
IV. Implementation

A. Vấn đề gặp phải

Trong quá trình giải bài toán, nhóm chúng em nhận thấy một vấn đề khó nhằn nảy sinh khi lấy mẫu kinh nghiệm từ từng tập. Cụ thể, khi epsilon giảm dần, tác tử chuyển từ giai đoạn khám phá sang khai thác, dẫn đến việc hành động sai lầm học được từ những tập trước tiếp tục ảnh hưởng đến hành vi của các tập sau. Điều này không chỉ làm giảm hiệu quả học tập mà còn tạo ra sức ỳ lên quá trình tối ưu hóa chiến lược. Về vi mô thì vấn đề này có thể không quá to tát khi kết quả thu được vẫn chấp nhận được.. Xét trên bối cảnh rộng hơn, MRDL ngày nay được ứng dụng trong nhiều lĩnh vực nhạy cảm như xe tự hành, hệ thống robot hậu cần tự động, UAV, ... thì việc xuất hiện dù chỉ một sai lầm nhỏ cũng có thể đem lại hậu quả rất lớn, cả về người và của. Điều này đặc biệt cần lưu tâm khi mô hình được ứng dụng vào các hệ thống có quy mô lớn. Do đó nhóm đã bắt tay vào tìm cách khắc phục và giải quyết triệt để vấn đề.

B. Khắc phục

Nhằm khắc phục hạn chế này, chúng em đã thay đổi cách tối ưu hóa mô hình. Thay vì thực hiện quá trình tối ưu hóa theo chu kỳ thông thường, nhóm đã điều chỉnh mô hình liên tục ngay sau mỗi bước hành động của tác tử trong quá trình huấn luyện. Cách tiếp cận này được kỳ vọng đem lại khả năng thích nghi nhanh hơn mô hình, đồng thời làm "mịn"đi khả năng kế thừa sai sót từ những tập dữ liệu trước đó. Do batch size có kích cỡ 128 mà



Hình 4. (a).Mạng trộn với phần màu đỏ là Hypernetwork. (b).Cấu trúc chung của qmix. (c) Mạng tác tử.

max cycle là 300 cho mỗi agent cho nên quá trình tối ưu diễn ra 172 lần trong tập 1 mà kinh nghiệm thì không mất nên mô hình tối ưu 300 lần mỗi tập trong những tập sau đó. Đây là một sự đánh đổi khi thời gian huấn luyện bị chậm đi đáng kể nhưng kết quả thu được đã tốt hơn mong đợi.

C. Phần thưởng

Trong dự án này, phần thưởng cũng đã được điều chỉnh để phù hợp với phương pháp huấn luyện, cụ thể: step reward = 0.01 để khuyến khích việc di chuyển thay vì để giá trị âm như mặc định, tăng attack opponent reward từ 0.2 lên 2 để khuyến khích tác tử tấn công, tăng phần thưởng khiến tác tử không còn sợ bị trừ điểm từ attack penalty = -0.1 cho những hành động tấn công không trúng. Mỗi hành động đều được gắn phần thưởng tạo nên môi trường dense reward, khiến quá trình hội tụ diễn ra nhanh hơn. Nhưng cũng cần cẩn thận vì thiết kế phần thưởng sai sẽ dẫn đến những hậu quả, hành vi không mong muốn của tác tử. Max cycles của mỗi tác tử cũng được đặt thành 300 để đảm bảo tác tử có thể tối ưu phần thưởng trong giới hạn thời gian mà đề bài đã cho.

D. QMIX: Thí nghiệm bên lề

QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning, là một thuật toán học tăng cường đa tác tử (MARL) dựa trên giá trị huấn luyện chính sách phi tập trung dựa trên quá trình học tập trung (centralized learning with decentralized execution) có khả năng nổi bật là khả năng ước lượng giá trị hành động chung (joint action-value) bằng việc kết hợp hành động riêng của các tác tử, với ràng buộc đảm bảo tính đơn điệu (monotonicity). Trong các bài toán phức tạp như StarCraft II, QMIX đã chứng minh hiệu quả vượt bậc khi so sánh với các thuật toán khác (Rashid, T., Samvelyan, M., De Witt, C. S., Farquhar, G., Foerster, J., & Whiteson, S. (2020). Monotonic value function factorisation for deep multi-agent reinforcement learning. Journal of Machine Learning Research, 21(178), 1–51.) [2].

1) Kiến trúc mô hình: Qmix sử dụng 3 mạng riêng biệt, cụ thể: mạng tác tử (Agent Network), HyperNetwork, mạng trộn (Mixing Network) (Hình 4).

a) Mạng Tác tử (Agent Network): Với tác dụng tạo ra q value của hành động, mạng tác tử nhận đầu vào là quan sát và id của tác tử, bao gồm các lớp tích chập để xử lý quan sát của tác tử và một lớp fully connected để tính toán Q-value cho từng hành động. Đầu tiên là 3 lớp tích chập (Conv1, Conv2, Conv3) được sử dụng để trích xuất đặc trưng từ quan sát của tác tử. Tương tự như DDQN, sau mỗi lớp tích chập dữ liệu đều được áp dụng hàm ReLU. ID của tác tử được chuyển thành một one-hot vector và cho đi qua một lớp fully connected để tạo ra một vector nhúng, giúp phân biệt các tác tử (được chứng minh là hiệu quả trong bài báo trên [2]). Sau đó vector nhúng của ID và kết quả của lớp tích chập được kết hợp với nhau rồi đi qua một lớp fully connected nữa để tính toán q value cho mỗi hành động.

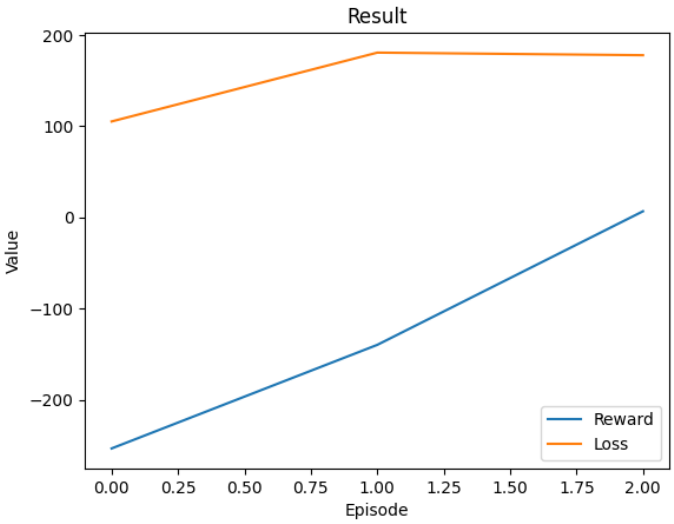
b) HyperNetwork: Mạng này có tác dụng tạo ra trọng số cho mạng trộn dựa trên trạng thái hiện tại của môi trường. Hai lớp tích chập (Conv1, Conv2) có vai trò trích xuất đặc trưng không gian từ trạng thái đầu vào sau đó sử dụng hàm kích hoạt ReLU. Sau khi trích xuất đặc trưng sử dụng lớp fully connected để tạo ra trọng số và bias cho mạng trộn.

c) Mạng trộn (Mixing Network): Có vai trò kết hợp các q value của các tác tử để tính toán giá trị tổng hợp. Nhận vào trạng thái chung (state) và q value của các tác tử. Cấu trúc mạng gồm: ba HyperNetwork để tạo trọng số và bias cho hai lớp trộn (w1, b1, w2, b2) từ trạng thái chung. Nhân các q values cho bộ tham số w1, b1 sau đó dùng hàm kích hoạt ELU, tiếp tục nhân với w2, b2 để cho ra kết quả cuối cùng là q value tổng hợp đại diện cho kết quả thể hiện của cả đội.

2) Phương pháp huấn luyện: Phương pháp huấn luyện của QMIX gần giống với DQN khi dùng chung Epsilon-Greedy Policy, Replay Buffer, hàm Loss và hàm tối ưu. Chỉ có một chút khác biệt về cách tối ưu khi phải cho tất cả tác tử hành động xong một lượt, lưu kinh nghiệm đó vào bộ nhớ riêng cho từng tác tử theo từng lượt và tính toán tất cả q value cùng lúc, sau đó tính toán q value chung hiện tại và q value chung mục tiêu và tính toán loss như bình thường. Quá trình huấn luyện cũng cần cài đặt thêm một số thứ như lưu trạng thái bắt đầu và kết thúc của từng lượt (mỗi lượt hoàn thành khi duyệt qua tất cả tác tử của một đội).

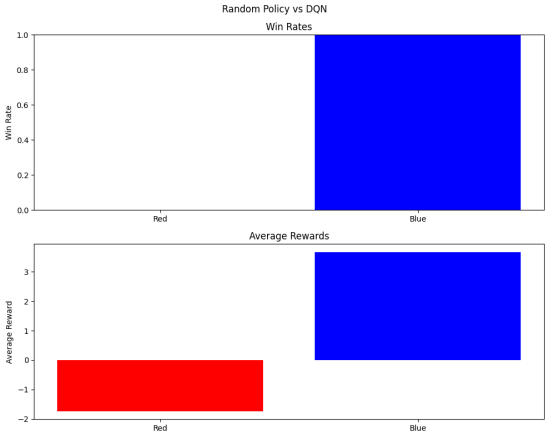
V. Experiment Results

a) **Đo lường hiệu suất** Đánh giá kết quả với tác tử được cho trước bên đỏ (red), tác tử được huấn luyện bằng DDQN ở bên xanh (blue)
Kết quả của quá trình huấn luyện: Losses và điểm thưởng theo từng episode:

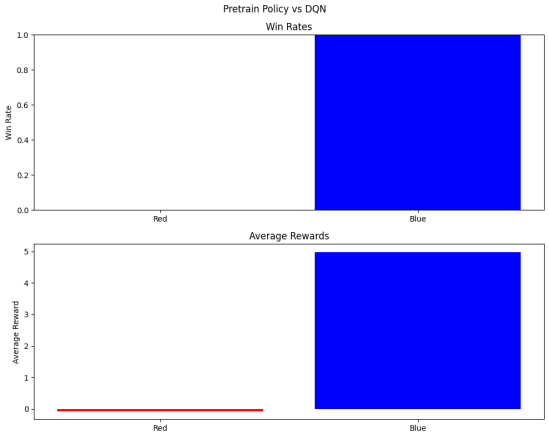


Hình 5. Performance metrics.

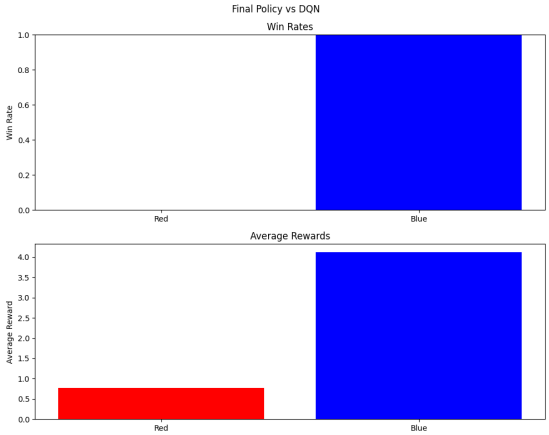
Kết quả của quá trình đánh giá:



Hình 6. Đánh giá với tác tử random.



Hình 7. Đánh giá với tác tử pretrain0.



Hình 8. Đánh giá với tác tử new pretrain.

Mặc dù Qmix đã được thử nghiệm và vận hành thành công nhưng do thời gian có hạn nên nhóm chưa thể đạt được kết quả mong muốn, có thể do siêu tham số hoặc quá trình huấn luyện mắc lỗi dẫn đến những hành động không tối ưu của tác tử nên kết quả thử nghiệm này sẽ không báo cáo ở đây.

b) **Thảo luận về tham số**

- **Epsilon**: ϵ_{start} = 1.0, ϵ_{end} = 0.1, ϵ_{decay} = 50: Khuyến khích khám phá mạnh ở giai đoạn đầu, giảm dần và chuyển sang khai thác ở giai đoạn sau.
- **Kích thước batch**: BATCH_SIZE = 128 đảm bảo dữ liệu huấn luyện đủ, đa dạng, cân bằng giữa tốc độ tính toán và bộ nhớ.
- **Hệ số chiết khấu:** γ = 0.9 cân bằng giữa phần thưởng ngắn hạn và dài hạn, vì môi trường khá đơn giản nên không có nhiều mục tiêu dài hạn.
- **Tốc độ học:** α = 1e-4 giúp cập nhật mạng ổn định, tránh dao động hơn so với 1e-3, tuy nhiên tốc độ huấn luyện sẽ chậm hơn.
- **Hệ số cập nhật mạng mục tiêu**: τ = 0.005 cập nhật mạng mục tiêu mượt mà, tránh thay đổi đột ngột vì quá trình cập nhật diễn ra liên tục sau mỗi hành động của tác tử.
- **Bộ nhớ kinh nghiệm**: Bộ nhớ với dung lượng lớn 10000 transitions, đảm bảo lưu trữ đủ mẫu từ nhiều trạng thái.
- **Số tập huấn luyện**: Chỉ cần 4 tập là đủ để mô hình cho ra kết quả tốt vì quá trình cập nhật mạng diễn ra liên tục và nhiều lần trong một tập, cần tăng khi môi trường phức tạp hơn.

c) **Thời gian huấn luyện**

Nhóm em sử dụng thuật toán DQNN huấn luyện xong 4 tập trong khoảng 1h30p, kết quả nhận được khá tốt. Nhóm cũng đã thử huấn luyện trong 12 giờ nhưng cũng chỉ đạt được kết quả tương tự (đôi khi tốt hơn do mất ít tác tử hơn).

VI. Conclusion

Nhóm đã áp dụng thành công Double Deep Q Network (DDQN) để giải quyết bài toán học tăng cường đa tác tử trong môi trường đối kháng - hợp tác. Kết quả huấn luyện và đánh giá sau 4 tập huấn luyện trong thời gian 1h30 phút đã thể hiện tính hiệu quả và ổn định của DDQN đối với những tác vụ phức tạp trong môi trường động. Ngoài ra, sự góp mặt của Replay Buffer, soft update cho mạng mục tiêu và chiến lược epsilon-greedy đã đảm bảo quá trình huấn luyện ổn định, giảm thiểu khả năng sai lệch của dữ liệu và tăng khám phá của mô hình. Những điều chỉnh về tham số, chẳng hạn như learning rate, discount factor và kích thước bộ nhớ, cũng đóng vai trò quan trọng trong việc đảm bảo hiệu suất mô hình được tối ưu.

Tuy nhiên, nếu bài toán có môi trường phức tạp hơn (Ví dụ: StarCraft 2), cần có sự điều chỉnh cấu trúc mạng hoặc áp dụng các kỹ thuật tăng cường khác nhằm nâng cao khả năng tổng quát hóa và giảm thiểu nguy cơ overfitting. Nhóm sẽ tiếp tục thử nghiệm và hoàn thiện bài toán bằng cách sử dụng q-mix, với mục tiêu tìm ra một lời giải tốt hơn, hay một chiến thuật hay hơn.

Tài liệu

[1] Matthew Hausknecht **andothers**. "Half field offense: An environment for multiagent learning and ad hoc teamwork". **in***AAMAS Adaptive Learning Agents (ALA) Workshop*: **volume** 3. sn. 2016.

[2] Tabish Rashid **andothers**. "Monotonic value function factorisation for deep multi-agent reinforcement learning". **in***Journal of Machine Learning Research*: 21.178 (2020), **pages** 1–51.

[3] Peter Stone, Richard S Sutton **and** Gregory Kuhlmann. "Reinforcement learning for robocup soccer keepaway". **in***Adaptive Behavior*: 13.3 (2005), **pages** 165–188.

[4] Gabriel Synnaeve **andothers**. "Torchcraft: a library for machine learning research on real-time strategy games". **in***arXiv preprint arXiv:1611.00625*: (2016).

[5] Oriol Vinyals **andothers**. "Starcraft ii: A new challenge for reinforcement learning". **in***arXiv preprint arXiv:1708.04782*: (2017).