Report for AI3007: Group 4

1st Nguyễn Đức Minh
22022533

2st Trần Tiến Nam
22022594

3st Phan Văn Hiếu
22022527

***Tóm tắt nội dung*—Báo cáo kết quả bài toán huấn luyện đa tác nhân trong môi trường Magent2 Battle. Môi trường Magent2 Battle tập trung vào việc tạo ra các chiến lược tương tác giữa nhiều tác nhân trong một không gian chiến đấu. Báo cáo này nhóm em sẽ tập chung chính vào thuật toán Deep Q Learning. Kết quả đã tạo ra được một mô hình có khả năng đánh bại cả 3 đối thủ (ngẫu nhiên và 2 mô hình được cho trước) với winrate đều là 1.0.**
Mô hình có thể được reproduce bằng cách chọn "Run All" trong notebook và model sẽ được lưu tự động dưới tên "best_model.pt", hàm eval() sẽ được gọi để đánh giá hiệu suất. Model cuối cùng để nộp được đóng zip cùng với báo cáo này.

I. Introduction

Explain the main approach: Nhóm em sử dụng mô hình DQN, thiết kế thêm một reward pursuit (cộng điểm cho việc hành động dẫn tối thấy nhiều địch hơn ở observation). Để tránh việc overfitting nhóm giới hạn chỉ train khi buffer chứa tổng samples của khoảng 5 episodes và chỉ lấy ngẫu nhiên các samples ra train mỗi khi thêm 4 samples mới. Nhóm cũng tránh overfitting bằng việc break sớm, không train đủ 100 epochs như set up ban đầu, nhờ vậy rút ngắn được thời gian huấn luyện.
Summarize results: Nhóm em đã train ra được nhiều model với tỉ lệ thắng 100%. Nhưng nhóm gặp vấn đề không phải lúc nào train lại cũng reproduce được kết quả toàn thắng. Nhóm đã hạn chế việc ngẫu nhiên bằng cách đặt seed cho môi trường, numpy, cuda torch nhưng vẫn không thể tránh được hoàn toàn vấn đề ngẫu nhiên. Train script bọn em nộp đã khá ổn định, khi train đa số sẽ ra kết quả toàn thắng, nhưng vẫn có khả năng model không thắng được final.
Link video và models: Link Driver

Bảng I
Main results

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

Average score for each models, using 1 point for Win, 0.5 point for Draw and 0 point for Lose:

- Random: 1
- Pretrain-0: 1
- New Pretrain: 1

II. Methods

A. Deep Q-Learning

DQN là một mở rộng của thuật toán Q-learning cổ điển, trong đó thay vì sử dụng bảng Q-table để lưu trữ giá trị Q cho từng trạng thái-hành động, DQN sử dụng một mạng nơ-ron sâu để xấp xỉ giá trị Q. Điều này làm cho DQN phù hợp với huấn luyện môi trường Magen2 Batte, do có thể giải quyết các bài toán có không gian trạng thái lớn hoặc liên tục, nơi mà Q-table truyền thống không khả thi.
DQN sử dụng một mạng nơ-ron Q(s,a;θ) để dự đoán giá trị Q tương ứng với mỗi trạng thái s và hành động a, với θ là các trọng số của mạng.
Cập nhật hàm Q: Trọng số θ được cập nhật bằng cách giảm hàm mất mát (loss function) dựa trên Bellman Equation. Nhóm bọn em cũng đã tìm hiểu về seminar của bên tạo ra môi trường và kết quả train của họ cho thấy DQN là hiệu quả nhất trong các thuật toán được sử dụng [1]. Đây cũng là một lý do để chúng em tập trung việc huấn luyện bằng DQN.
Link bài báo: MAgent: A Many-Agent Reinforcement Learning Platform for Artificial Collective Intelligence.
Nhóm sử dụng hai mạng riêng biệt là Q-Network và Target Q-Network để cập nhật trọng số một cách hiệu quả. Q-Network đóng vai trò xấp xỉ hàm giá trị Q-value cho các cặp (state, action), từ đó dự đoán giá trị kỳ vọng của phần thưởng trong tương lai. Target Q-Network được sử dụng để tính Q-value mục tiêu trong quá trình cập nhật Q-Network. Target Q-Network có các trọng số được cố định tạm thời và chỉ được cập nhật từ Q-Network sau mỗi 10 episodes. Điều này giúp giảm sự dao động trong mục tiêu của hàm mất mát, từ đó ổn định quá trình huấn luyện và cải thiện khả năng hội tụ của mô hình.

B. Lấy trải nghiệm

Môi trường battle_v4 cung cấp hàm lấy thông tin của mỗi tác nhân:
obervation, reward, temination, truncation, info = env.last()
Tuy nhiên, sau khi agent thực hiện hành động tại iter hiện tại, reward sẽ không được cập nhật ngay lập tức mà phải đợi đến iter tiếp theo của agent đó. Để giải quyết vấn đề này, chúng em có tạo thêm một dictionary tạm thời để lưu trữ các thông tin (observation, action, next_observation). Khi bước sang iter tiếp theo, chúng em bổ sung reward và done (done = termination or truncation) rồi hoàn thiện bộ dữ liệu (observation, action, reward, next_observation, done) để đưa vào Replay Buffer.

C. Replay Buffer

Replay Buffer là một thành phần thiết yếu trong các thuật toán học tăng cường sâu như DQN. Nó lưu trữ các kinh nghiệm

mà tác nhân thu được trong quá trình tương tác với môi trường, bao gồm các thông tin như state, action, reward, next state và done. Replay Buffer thường được thiết kế với kích thước chiếm khoảng 30-50% tổng số samples sinh ra trong quá trình huấn luyện, vừa đủ lớn để lưu trữ những trải nghiệm phong phú cho tác nhân học tập, vừa đủ nhỏ để loại bỏ kịp thời những trải nghiệm quá cũ.
Việc lấy trải nghiệm ngẫu nhiên giúp tác nhân tiếp cận được một tập hợp các trải nghiệm đa dạng, từ đó tránh việc lặp đi lặp lại một hành động duy nhất quá nhiều lần, giúp mô hình học hỏi được sự phong phú của môi trường và cải thiện khả năng tổng quát.

D. Reward

Nhóm chúng em đã thử tạo thêm reward và điều chỉnh reward gốc để có thể tạo ra nhiều chiến thuật khác nhau:

- Do hành động tấn công thành công và hạ gục kẻ địch rất ít so với hành động di chuyển và tấn công thất bại nên chúng em đã tăng reward lên thành 0.5 để khuyến khích agents tấn công.
- Khi một blue_agent chết đi, reward sẽ giảm dựa trên số red_agent còn sống, tăng dựa trên red_agent đã chết.
- Khi agent thực hiện một hành động, chúng em sẽ so sánh giữa số lượng quân địch ở state và next state. Nếu số lượng quân địch ở next state đông hơn, chúng em sẽ thưởng cho hành động đó.
- Khi agent thực hiện một hành động, nếu ở cả state và next state đều có hiện diện của địch, chúng em sẽ tính trung bình khoảng cách euclid, nếu khoảng cách ở next state nhỏ hơn thì thưởng cho agent (khuyễn khích agent tiến đến đối thủ khi mô phỏng để giảm bớt khả năng quá nhiều dữ liệu vô nghĩa khi train với số episodes không lớn).

E. Lưu samples

Như đã đề cập trước đó, tỷ lệ các hành động có reward dương, chẳng hạn như tấn công thành công hoặc hạ gục, là rất nhỏ so với các hành động còn lại trong môi trường ban đầu. Để đảm bảo tính đa dạng cho các samples trong buffer và tránh tình trạng các hành động có reward âm chiếm ưu thế, nhóm đã nhân bản các samples này nhiều lần (gấp đôi hoặc gấp ba) trước khi đưa vào buffer. Nhóm cũng đã phân tích cụ thể hành vi của agent và nhận thấy rằng các hành động 4 và 5 khiến agent tiến thẳng vào trung tâm đội hình quân địch của đội xanh. Vì vậy, nhóm chúng em cũng tiến hành nhân bản các samples chứa hai hành động này để tăng tần suất xuất hiện của chúng trong buffer.

F. Exploratory policy

Chúng em áp dụng chính sách epsilon-greedy để điều khiển quá trình khám phá của agent. Cụ thể, với một giá trị ngẫu nhiên được khởi tạo, nếu giá trị này nhỏ hơn hệ số epsilon, agent sẽ thực hiện một hành động ngẫu nhiên; ngược lại, agent sẽ chọn hành động có Q-value cao nhất được dự đoán bởi Q-Network. Hệ số epsilon ban đầu được đặt là 1 và sẽ giảm dần theo công thức:

epsilon = max(final_epsilon, initial_epsilon * math.exp(-decay_rate * episode))

Việc giảm dần epsilon giúp agent cân bằng giữa việc khám phá môi trường và khai thác các hành động tối ưu đã học được.

G. Dừng huấn luyện sớm

Ban đầu, nhóm chúng em thiết lập quá trình huấn luyện với 100 episodes. Tuy nhiên, khi quan sát giá trị return, nhóm nhận thấy mô hình có dấu hiệu overfitting ở các vòng lặp cuối. Do đó, chúng em quyết định dừng sớm quá trình huấn luyện nhằm tránh tình trạng overfit và tiết kiệm tài nguyên tính toán.

III. Implementation

Đầu tiên nhóm em khởi tạo hai model q_network và target_q_network, q_network dùng để sinh q_values còn target_q_network sẽ sinh next_q_values. target_q_network được cập nhập mỗi 10 episodes để đảm bảo tính ổn định cho model. Buffer size nhóm em để 50000 và sẽ chỉ bắt đầu lấy samples ra train khi buffer size đạt tối thiểu 4800.
Nhóm đã thử với nhiều bộ hyperparameters và cuối cùng thu được một bộ hyperparameters phù hợp nhất. Bộ hyperparameters này như sau:

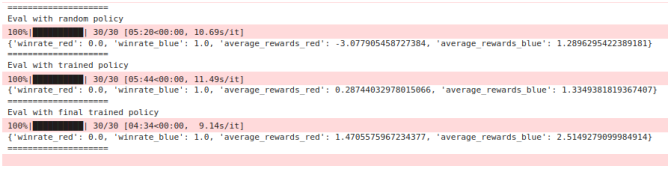
- batch_size = 16
- episodes = 100
- gamma = 0.9
- decay_rate = 0.02
- intial_epsilon = 1.0

Khi huấn luyện ta sẽ lưu observation, action, next observation của agent vào dictionary và khi đến iter sau sẽ lấy nốt hai biến done (do môi trường cập nhập chậm reward) để lưu vào buffer. Reward cũng sẽ được cộng thêm do hàm pursuit trả về, hàm sẽ trả về 0.015 nếu hành động của agent dẫn đến nhiều địch hơn cho next observation. Chúng em tìm số lượng địch bằng cách đếm số vị trí có giá trị lớn hơn 0 trong channel "hp kẻ địch". Vì muốn khuyến khích agent đi lên tấn công nên chúng em chỉ reward của hàm pursuit cho các hành động tiến lên hay đi sang ngang, các hành động lùi xuống (3,7,8,11,12,6,0,2,10) sẽ không được cộng.
Vì muốn ưu tiên các hành động tấn công hay di chuyển về phía địch nên khi gặp nhưng hành động có giá trị lớn hơn 13 hay các hành động tiến thẳng về phía trước (4,5) bọn em sẽ nhân bản 2 lần sample này để đưa vào buffer.
Khi điều kiện buffer size đạt được, model sẽ lấy 16 samples từ buffer, q_value sẽ được tính bằng observation và chọn theo action (dùng hàm gather(action) của torch), next_value sẽ được tính bằng next_observation, từ đó ta tính được target. Ta sẽ tính loss theo mean square error giữa q_value và target. Sau khi tính loss sẽ thực hiện cập nhập trọng số, nhóm sử dụng optimizer Adam của torch với learning rate = 0.0001. Do để tránh overfitting nhóm em thêm một điều kiện là buffer có thêm 4 samples mới thì mới cập nhập.
Sau khi thử nghiệm nhiều thì nhóm em thấy được model sẽ có kết quả tốt ở khoảng episode 25-35. Do vậy chúngem đã break quá trình training ngay ở episode 35. Chúng em đã lưu

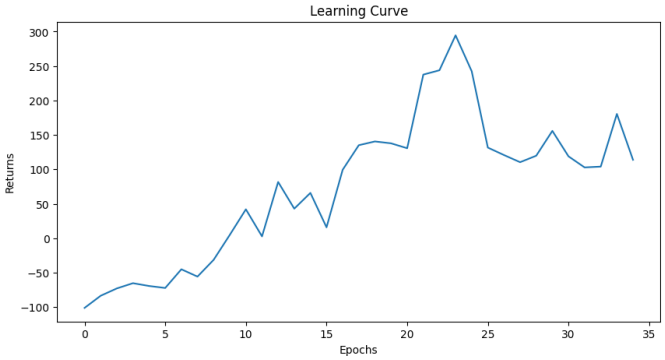
lại model tốt nhất bằng cách lưu lại model giết được nhiều địch nhất.

IV. Experiment Results

Sau khi ổn định lại mô hình, chúng em đã đưa ra được mô hình với winrate như Hình 1



Hình 1. Winrate



Hình 2. Biểu đồ giá trị ep_reward theo episode.

Biểu đồ như Hình 2 cho thấy được agent đã biết cách tấn công, di chuyển cũng như đã giết được một số lượng địch nhất định để thu được giá trị ep_return (tổng reward của toàn bộ blue_agent) dương như vậy. Reward theo hình cao như vậy vì đây là biểu đồ vẽ theo reward đã được thêm điểm thưởng của hàm pursuit

Chúng em đã thử training lại model để kiểm tra tính ổn định. Tuy nhiên train script cũng có một khả năng nhỏ ra kết quả hơi kém (không thắng dược final).

Thời gian training trung bình khoảng 10-15 phút trên google colab.

V. Conclusion

Như vậy, nhóm đã thực hiện huấn luyện thành công mô hình có thể đánh bại cả 3 mô hình cho trước (random, pretrained và final) với chi phí tính toán khá thấp. Điều này cho thấy sức mạnh của thuật toán Deep Q-learning không chỉ cho các bài toán đơn tác tử mà còn cho cả các bài toán đa tác tử. Mặc dù gặp khó khăn trong việc dễ rơi vào tối ưu cục bộ và thiếu tính khái quát hóa, nhóm đã sử dụng một số kỹ thuật giúp mô hình có hành vi tốt hơn mà không cần tăng quá nhiều tài nguyên tính toán. Trong tương lai, có thể nhóm sẽ thử huấn luyện với môi trường rộng hơn với nhiều tác tử hơn bằng các thuật toán khác.

Tài liệu

[1] Lianmin Zheng, Jiacheng Yang, Han Cai, Ming Zhou, Weinan Zhang, Jun Wang, and Yong Yu. Magent: A many-agent reinforcement learning platform for artificial collective intelligence. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.