BÁO CÁO BÀI TẬP LỚN
Môn học: Học tăng cường và lập kế hoạch

1st Bùi Tiến Sâm
22022517@vnu.edu.vn
Nhóm trưởng

2nd Nguyễn Văn Mạnh
22022521@vnu.edu.vn

3rd Dương Phương Hiếu
22022659@vnu.edu.vn

***Tóm tắt nội dung*—Bản báo cáo này, thực hiên bởi nhóm 5, sẽ nói về phương pháp để huấn luyện một mô hình học tăng cường trong môi trường là một trận đấu đối kháng giữa các tác tử với nhau. Môi trường được sử dụng trong bản báo cáo này là MAgent2: https://github.com/Farama-Foundation/MAgent2**

I. Giới thiệu sơ lược

Học tăng cường (Reinforcement Learning) là một nhánh của trí tuệ nhân tạo, với đặc trưng là các thuật toán học cách hành động trong một môi trường để tối ưu hóa một phần thưởng dài hạn. Học tăng cường với nhiều tác tử (Multi-Agent Reinforcement Learning - MARL) là ý tưởng mở rộng của học tăng cường đơn tác tử (Single-Agent) bằng cách tích hợp nhiều tác nhân hoạt động trong cùng một môi trường. Việc lựa chọn hành động của từng tác tử sẽ do mô hình học tăng cường quyết định. Và phương pháp để tạo ra mô hình đó là chủ đề chính của bản báo cào này.

Bản báo cáo này sẽ gồm 6 phần:

- Phần I, giới thiệu sơ lược về chủ đề của báo cáo.
- Phần II, giải thích ngắn gọn về cách hoạt động của môi trường Magent2.
- Phần III, trình bày về DQN, phương pháp học tăng cường chính được sử dụng trong bài toán này.
- Phần IV, đánh giá phương pháp đã trình bày và nêu kết quả đạt được.
- Phần V, trình bày về các phương pháp khác cũng đã được sử dụng
- Phần VI, tổng kết và hướng phát triển trong tương lai.

II. Môi trường MAgent2

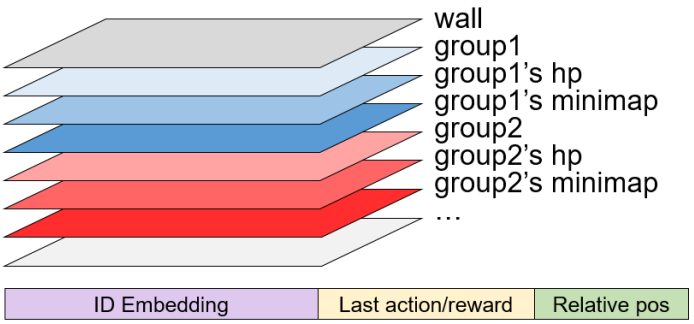
MAgent2 - môi trường mô phỏng đa tác nhân được sử dụng trong bản báo cáo này, là một thư viện dùng để tạo ra các môi trường nơi số lượng lớn các tác nhân dạng pixel trong một thế giới dạng lưới tương tác thông qua các trận chiến hoặc các tình huống cạnh tranh khác.

A. Tác tử

MAgent2 có nhiều phiên bản mội trường cài đặt khác nhau. Tuy nhiên, trong bản báo cáo này, chúng tôi lựa chọn một dạng môi trường đơn giản của MAgent, là môi trường đấu đối kháng giữa 162 tác tử, được chia thành 2 đội, xanh và đỏ, trong một khoảng diện tích 45x45 pixel, không có vật cản. Mỗi tác tử của mỗi đội đều có một lượng Hit Point (HP) nhất định. Giống như trong những trò chơi bình thường, khi lượng HP này giảm xuống con số 0, tác tử được coi là đã bị tiêu diệt và sẽ biến mất khỏi môi trường hiện tại. Khỏi đầu trò chơi, mỗi tác tử đều có 10HP, mỗi lần tác tử bị tấn công, lượng HP sẽ giảm xuống 2 đơn vị, song sau mỗi bước di chuyển, tác tử sẽ được tự động hồi lại 0.1HP.

B. Trạng thái

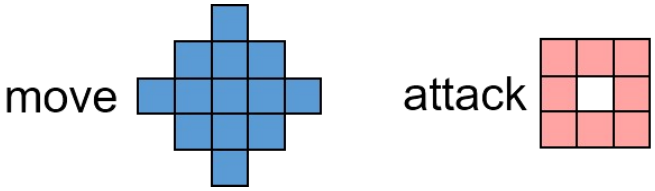
Trạng thái hiện tại của môi trường được đặc trưng bởi tầm nhìn (observation) của các tác tử. Mỗi tác tử đều có được một tầm nhìn là một hình vuông 13x13, chứa các thông tin về vị trí hiện tại của bản thân tác tử, đồng đội, đối thủ và cả lượng HP còn lại, tất nhiên là nếu có tồn tại tác tử đồng đội hoặc đối thủ nằm ở trong tầm nhìn.



Hình 1. Mô tả một tầm nhìn của tác tử

C. Hành động

Mỗi tác tử đều có thể thực hiện một trong 21 hành động ở mỗi lượt, bao gồm 13 hành động "di chuyển" (tính cả "đứng yên" được biểu diễn bởi 0 là một hành động) và 8 hành động "tấn công".



Hình 2. Mô tả hành động của tác tử

D. Phần thưởng

Một đặc trưng của bài toán học tăng cường khác với những bài toán khác thuộc lĩnh vực trí tuệ nhân tạo chính là hệ thống phần thưởng. Bằng cách tối đa hoá lượng phần thưởng nhận được trong dài hạn, ta sẽ dạy được tác tử cách để đạt được mục tiêu mong muốn.

Hệ thống phần thưởng được sử dụng trong bài toán này được liệt kê như sau:

- 5 điểm mỗi lần tiêu diệt được đối thủ.
- -0.005 điểm sau mỗi bước di chuyển của tác tử.
- -0.1 điểm sau mỗi lần tác tử chọn hành động "tấn công".
- 0.2 điểm mỗi lần đối tượng bị tác tử tấn công thuộc đội đối thủ.
- -0.1 điểm khi tác tử bị tiêu diệt.

E. Mục tiêu của bài toán

Mục tiêu của bài toán là tối đa hóa phần thưởng tích lũy hay nói cách khác là giành được chiến thắng tuyệt đối. Mỗi tác tử sẽ học cách hành động bằng phương pháp sẽ được đề cập ở phần dưới để tiêu diệt nhiều tác tử đối thủ nhất có thể. Ở cuối trận chiến, đội nào còn nhiều tác tử còn sống hơn thì được coi là chiến thắng.

III. Phương pháp DQN

Ý tưởng chính của DQN là kết hợp mạng Deep Neural Network và thuật toán Q-Learning. Thay vì giá trị của mỗi hành động, trạng thái được lưu trong một Q-table thì nó được lưu bằng trọng số của mô hình DQN.

A. Sơ lược về DQN

1) Q-Learning truyền thống sử dụng Q-table: Trong các bài toán học tăng cường, **action-value function** thường được sử dụng để ước tính hành động nào là tối ưu cho từng trạng thái cụ thể. Một trong những phương pháp phổ biến là **Q-Learning**, với các giá trị của mỗi cặp trạng thái-hành động (s, a) được học thông qua các phép cập nhật lặp đi lặp lại. Tuy nhiên, phương pháp này phải đối mặt với một thách thức lớn: không gian trạng thái-hành động (s, a) thường rất lớn, đòi hỏi lượng tài nguyên lưu trữ khổng lồ để có thể bao phủ hết toàn bộ không gian này. Đây là một nhược điểm đáng kể, đặc biệt trong các môi trường có độ phức tạp cao hoặc không gian trạng thái liên tục.

2) Tham số hóa giá trị Q: Để khắc phục nhược điểm trên, việc **tham số hóa** trở thành một giải pháp tối ưu. Thay vì lưu trữ toàn bộ giá trị (s, a) trong bảng tra cứu (lookup table), các giá trị này được "lưu trữ" bởi một hàm số với các tham số tiềm năng, chẳng hạn như mạng nơ-ron. Mô hình tham số hóa không chỉ tiết kiệm bộ nhớ mà còn giúp mở rộng khả năng tổng quát hóa, cho phép ước lượng giá trị của các trạng thái chưa từng gặp thông qua quá trình học.

Phương pháp này trở nên rất hữu ích trong các bài toán có không gian trạng thái liên tục hoặc môi trường phức tạp, trong đó các phương pháp dựa trên bảng trở nên bất khả thi. Với cách tiếp cận tham số hóa, hệ thống có thể dự đoán giá trị của một trạng thái đầu vào bất kỳ một cách nhanh chóng và chính xác.

B. Thuật toán DQN

Trong DQN, **Q-value Estimation** là một trong những bước cốt lõi để đánh giá giá trị của một hành động cụ thể tại một trạng thái. Công thức cập nhật Q-Learning dựa trên sai số **TD Error**, thể hiện khoảng cách giữa giá trị Q ước tính hiện tại $Q(s, a)$ và mục tiêu TD Target. Trong quá trình huấn luyện, mục tiêu là làm cho sai số TD nhỏ dần, sao cho $Q(s, a)$ hội tụ về giá trị mục tiêu. Khi tham số hóa các giá trị Q, mục tiêu của chúng ta trở thành việc tìm tham số θ để cực tiểu hóa hàm mất mát $(y_j - Q(\phi_j, a_j; \theta))^2$, trong đó y_j là TD Target được tính toán từ mạng cố định θ^-. Phương pháp tham số hóa này không chỉ giúp mô hình ổn định hơn mà còn tăng khả năng tổng quát hóa, mở rộng phạm vi áp dụng của thuật toán.

C. Phương pháp huấn luyện mô hình

Để tối ưu hóa thuật toán **Deep Q-Learning (DQN)**, quá trình huấn luyện thường được chia thành hai giai đoạn chính:

1) Sampling dữ liệu:: Ở giai đoạn này, mô hình thực hiện các hành động và lưu trữ trải nghiệm dưới dạng các tuple (s, a, r, s', a'). Một vấn đề thường gặp là nếu sử dụng dữ liệu theo thứ tự thời gian, mô hình có nguy cơ bị fit vào các phụ thuộc này, làm giảm khả năng học các đặc trưng của trạng thái. Để giải quyết, chúng tôi sử dụng **Replay Memory** để lưu trữ trải nghiệm và lấy các batch dữ liệu ngẫu nhiên, đảm bảo tính độc lập giữa các mẫu trong quá trình huấn luyện.

2) Training mô hình:: Giai đoạn này tập trung vào việc cập nhật mạng Q bằng cách sử dụng kỹ thuật **Fixed Q-Target** để cải thiện sự ổn định. Một vấn đề nổi bật trong Q-Learning truyền thống là việc cả giá trị Q và mục tiêu TD đều được ước tính từ cùng một mạng, dẫn đến dao động lớn trong quá trình huấn luyện. Đây giống như việc đuổi theo một mục tiêu đang di chuyển, khiến việc hội tụ trở nên khó khăn. Để khắc phục, chúng ta sử dụng một mạng Q mục tiêu cố định với tham số θ^- để tính toán TD Target. Sau mỗi C bước, tham số từ mạng Q chính được sao chép để cập nhật mạng mục tiêu. Cách tiếp cận này không chỉ giảm thiểu sự dao động mà còn đảm bảo quá trình huấn luyện hội tụ ổn định hơn.

IV. Thí nghiệm

A. Mô hình sử dụng

Trong thí nghiệm này, chúng tôi sử dụng một mô hình DNN (Deep Neural Network) với hai thành phần chính:

1) Convolution Neural Network (CNN):: Đặc trưng của mạng tích chập là trích xuất các đặc trưng sử dụng cửa sổ trượt. Ở đây chúng tôi sử dụng 3 lớp Conv2d, tất cả các lớp đều cho ra 13 channels. Ở 2 lớp đầu tiên, chúng tôi sử dụng padding và stride để giữ nguyên kích thước so với đầu vào. Ở lớp conv2d cuối, chúng tôi sử dụng padding 0 làm cho kích thước của đầu ra giảm đi. Sau khi flatten, những lớp conv2d đó khiến cho lượng features không quá ít (underfitting) và cũng không quá nhiều (overfitting). Mỗi lớp conv2d đều được gắn với một hàm ReLU để phi tuyến tính hóa.

2) Fully Connected Network (FC):: Mạng fully connected được thiết kế nhằm xử lý dữ liệu đã được flatten từ các bước trước đó. Chúng tôi sử dụng một cấu trúc tuần tự gồm ba lớp dense. Lớp đầu tiên có kích thước đầu vào là $flatten\_dim$ và đầu ra là 256 neurons, kết hợp với hàm kích hoạt ReLU để phi tuyến tính hóa. Lớp thứ hai tiếp tục giảm số chiều đầu ra xuống còn 128 neurons, cũng sử dụng hàm ReLU. Cuối cùng, lớp dense cuối cùng giảm số chiều xuống còn $action\_shape$, phù hợp với số lượng hành động cần dự đoán. Cách thiết kế này cho phép mạng học được các biểu diễn đặc trưng phức tạp trong không gian đầu ra, đồng thời giảm dần số chiều để tránh overfitting.

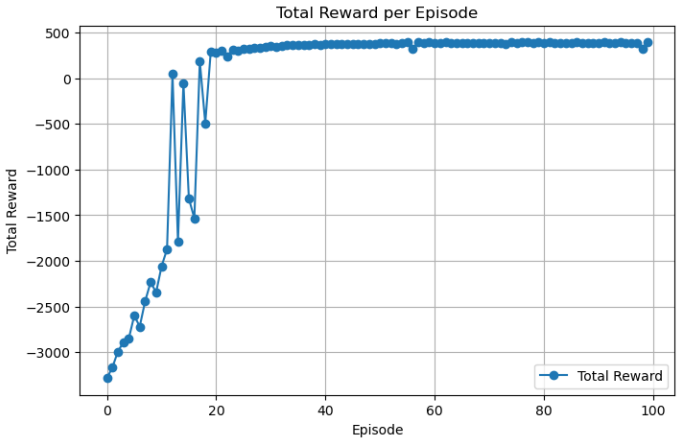
B. Quá trình huấn luyện

Chúng tôi huấn luyện mô hình DQN với 100 episodes trong vòng 13m26s bằng device GPU P100 của kaggle.

Sau mỗi một episodes, một chuỗi các trajactories được thêm vào replay buffer để thực hiện việc lưu trữ và sampling lại. Sau đó, một dataloader được rút ra từ replay buffer này để chuẩn bị cho việc huấn luyện. Tại mỗi episode, chúng tôi chỉ huấn luyện một lần trên một dataloader, tức là coi mỗi episode là một epoch.

Việc cập nhật các tham số do update target every đảm nhiệm.

C. Kết quả



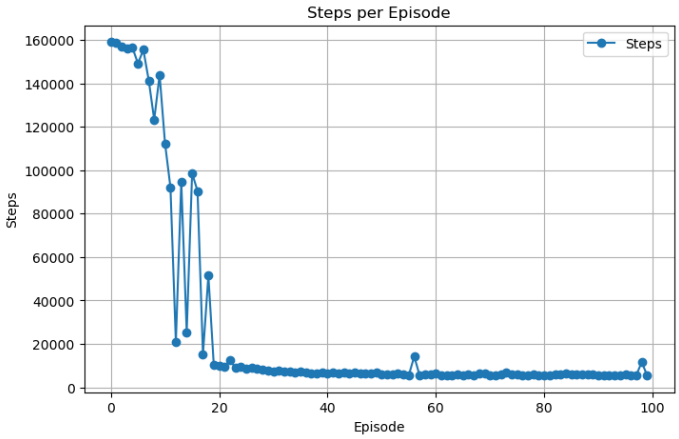
Hình 3. Tổng phần thưởng theo từng episode trong quá trình train

Policy	Winrate Red	Winrate Blue	Average Rewards Red	Average Rewards Blue
Red Final Policy	0.0	1.0	1.5335	4.8663
Red Policy	0.0	1.0	0.2403	4.9834
Random Policy	0.0	1.0	-0.9832	4.9292

Bảng I
KẾT QUẢ ĐÁNH GIÁ FINAL BLUE VỚI RANDOM, RED VÀ RED FINAL

D. Đánh giá kết quả

1) Trainning: Quá trình huấn luyện Blue đối đầu với Red (Random Policy) cho thấy sự tiến bộ rõ rệt thông qua các chỉ số như tổng phần thưởng (Total Reward) và số bước (Steps) cần thiết để hoàn thành mỗi trận đấu. Ban đầu, tổng phần thưởng của Blue rất âm, thể hiện sự non nớt trong chiến lược,



Hình 4. Tổng số bước di chuyển để hạ đối thủ theo từng episode trong quá trình train

nhưng dần cải thiện và chuyển sang giá trị dương từ Episode 12, đánh dấu bước ngoặt quan trọng khi Blue bắt đầu vượt trội hơn Red. Số bước thực hiện cũng giảm mạnh từ khoảng 159,000 xuống còn 6,000-7,000 bước, minh chứng cho chiến lược ngày càng hiệu quả và tinh gọn.

Quá trình huấn luyện có thể chia làm ba giai đoạn chính. Trong giai đoạn đầu (Episode 0-11), epsilon giảm từ 1.0 xuống 0.64 (epsilon bắt đầu từ 1 và giảm 0.96 lần sau mỗi episode) [1], là quá trình khám phá diễn ra mạnh mẽ. Dù tổng phần thưởng vẫn âm nhưng nó có xu hướng tăng cho thấy Blue vẫn đang học hỏi liên tục thông qua quá trình khám phá. Giai đoạn thứ hai (Episode 12-26) khi mà epsilon thấp dần chỉ còn khoảng 0.35 (chuyển dịch từ khám phá sang khai thác), chứng kiến bước nhảy vọt về hiệu suất từ ≈1700 đến 300 mặc dù vẫn còn giao động mạnh. Tổng phần thưởng tăng mạnh và ổn định ở mức dương, trong khi số bước giảm đáng kể. Cuối cùng, ở giai đoạn ba (Episode 27 trở đi) là giai đoạn khai thác khi mà epsilon giảm xuống 0.14. Trong giai đoạn này Blue đạt mức phần thưởng cao, ổn định và số bước duy trì thấp.

2) Evaluate: Kết quả đánh giá cho thấy Blue Agent đạt hiệu suất vượt trội khi đối đầu với tất cả các phiên bản của Red, từ chính sách cuối cùng (final policy), chính sách ban đầu (policy), đến chính sách ngẫu nhiên (random policy). Khi gặp Red sử dụng final policy - phiên bản tốt nhất, Blue đã giành chiến thắng tuyệt đối với tỷ lệ thắng 100% và phần thưởng trung bình 4.87, bỏ xa Red với chỉ 1.53 điểm. Điều này khẳng định năng lực chiến thuật và khả năng thích ứng xuất sắc của Blue ngay cả trong môi trường đối kháng khó khăn nhất.

Khi đối đầu với Red ở chính sách đầu tiên, Blue tiếp tục duy trì thế áp đảo với phần thưởng trung bình 4.98, trong khi Red chỉ đạt 0.24. Kết quả này thể hiện sự ổn định và khả năng khai thác điểm yếu đối thủ một cách hiệu quả, điều này khá rõ ràng khi chiến đấu với đối thủ red chưa có cải tiến nhiều về mặt kĩ thuật. Đặc biệt, khi đối mặt với Red random policy, Blue dễ dàng giành phần thắng với phần thưởng trung bình

[1]Các con số 0.64, 0.35, 0.14 được đề cập được tính bằng công thức: $0.96^{episode}$

4.93, trong khi Red bị âm (-0.98). Tất cả cho thấy Blue đã phát triển một chiến lược hoàn chỉnh và chắc chắn, bất kể đối thủ chơi ngẫu nhiên hay có chiến lược.

Kết quả đánh giá khẳng định rằng quá trình huấn luyện của Blue đã thành công vượt bậc. Blue không chỉ vượt trội khi đối đầu với chính sách ngẫu nhiên mà còn thể hiện khả năng thích ứng và khai thác chiến thuật hiệu quả trước các chính sách tối ưu hơn của Red. Sự nhất quán trong tỷ lệ thắng tuyệt đối và phần thưởng trung bình cao ở mọi trường hợp chứng minh rằng Blue đã phát triển chiến lược toàn diện, ổn định và hiệu quả.

V. Tổng kết và hướng phát triển

Kết quả của bài toán lần này cho thấy Blue Agent vượt trội hoàn toàn khi đối đầu với Red ở mọi phiên bản chiến lược. Blue duy trì tỷ lệ thắng 100% và phần thưởng trung bình cao, từ 4.87 đến 4.98, trong khi Red đạt điểm số thấp, thậm chí âm ở chiến lược ngẫu nhiên. Kết quả này khẳng định quá trình huấn luyện đã thành công, giúp Blue có thể chọn được hành động hợp lý trong môi trường đấu đối kháng.

Trong tương lai, chúng tôi có thể sẽ tăng độ phức tạp của mô hình học sâu hoặc thay đổi phương pháp huấn luyện để tìm được một giải pháp tốt hơn. Khi đó, kết quả của bản báo cáo lần này, mô hình chiến lược của Blue, sẽ trở thành đối thủ của phương pháp mới.

Dưới đây là link Kaggle của code huấn luyện và code đánh giá:

- Huấn luyện: train-battle-with-blue-agent
- Đánh giá: evaluate-battle-with-blue-agent