Xây dựng mô hình học tăng cường cho trò chơi Snake

Sinh viên:
Nguyễn Bình Nam
Đinh Nhật Thành
Ngô Đức Thuận

Giảng viên:
Đỗ Thanh Hà

Ngày Tháng Năm: Ngày 27 tháng 8 năm 2024

Mục lục

1 Giới thiệu bài toán **3**

2 Giới thiệu Học tăng cường **4**
2.1 Khái quát về học tăng cường 4
2.2 Mục tiêu bài toán 4
2.3 Môi trường 4
2.4 Tác nhân 5
2.5 Phần thưởng 5
2.6 Tại sao nên chọn học tăng cường cho bài toán 5

3 Xây dựng trò chơi sử dụng Deep Q-Learning **6**
3.1 Thuật toán Deep Q-Learning 6
3.2 Cấu trúc mạng Neuron 7
3.3 Tối ưu hóa thuật toán 8
3.4 Xây dựng trò chơi Snake AI 9
3.4.1 Xây dựng trò chơi Snake AI 9
3.4.2 Cấu trúc Tệp 10

4 Huấn luyện và đánh giá **12**
4.1 Huấn luyện thuật toán 12
4.2 Đánh giá thuật toán 13

5 Kết quả **15**
5.1 Quá Trình Huấn Luyện 15
5.2 Hiệu Suất Tổng Quát 15
5.3 Phân Tích Hiệu Suất 15
5.3.1 Điểm Số Tốt Nhất 15
5.3.2 Điểm Số Thấp Nhất 15
5.4 Phân Tích Biến Động 15
5.5 Kết Quả Cuối Cùng 16

6 Đề xuất cải thiện **17**
6.1 Mạng neuron 17
6.2 Tối ưu hóa mô hình 17

7 Kết luận **18**

Chương 1

Giới thiệu bài toán

Dự án này tập trung vào việc phát triển một mô hình học tăng cường cho trò chơi Snake, một trò chơi kinh điển mà trong đó người chơi điều khiển một con rắn di chuyển trong lưới với mục tiêu ăn thức ăn để kéo dài cơ thể và ghi điểm. Trò chơi yêu cầu người chơi phải liên tục đưa ra các quyết định nhằm tối ưu hóa điểm số, đồng thời tránh va chạm với tường hoặc chính cơ thể của mình. Mặc dù có vẻ đơn giản, trò chơi Snake lại đòi hỏi một chiến lược thông minh và tính toán kỹ lưỡng để đạt được thành công, đặc biệt khi con rắn ngày càng dài và không gian di chuyển càng bị thu hẹp.

Trong bối cảnh này, bài toán mà chúng tôi đặt ra là làm thế nào để xây dựng một tác nhân AI có thể tự học cách chơi trò chơi Snake một cách hiệu quả. Thay vì lập trình sẵn các chiến lược, chúng tôi áp dụng phương pháp học tăng cường, cho phép tác nhân học hỏi từ kinh nghiệm của chính mình thông qua tương tác với môi trường. Mỗi lần tác nhân thực hiện một hành động, nó sẽ nhận được phản hồi dưới dạng phần thưởng hoặc hình phạt, từ đó điều chỉnh hành vi để tối ưu hóa kết quả trong các lần chơi tiếp theo. Đây là một phương pháp học tập mạnh mẽ, giúp tác nhân có thể dần dần phát triển các chiến lược tối ưu mà không cần sự can thiệp trực tiếp của con người.

Việc áp dụng học tăng cường trong trò chơi Snake không chỉ giúp phát triển một tác nhân AI có khả năng chơi trò chơi này một cách xuất sắc, mà còn cung cấp những hiểu biết sâu sắc về cách thức mà AI có thể học từ môi trường phức tạp và động. Dự án này cũng sẽ giúp kiểm chứng hiệu quả của học tăng cường trong các tình huống thực tế, nơi mà môi trường không ngừng thay đổi và mỗi quyết định đều có thể dẫn đến những kết quả khác nhau.

Tóm lại, bài toán xây dựng mô hình học tăng cường cho trò chơi Snake đặt ra những thách thức đáng kể trong việc thiết kế và huấn luyện tác nhân AI. Thông qua dự án này, chúng tôi kỳ vọng sẽ không chỉ tạo ra một mô hình AI có khả năng chơi Snake thành công mà còn mở rộng ứng dụng của học tăng cường trong các lĩnh vực khác, nơi mà khả năng ra quyết định và tối ưu hóa hành động là yếu tố then chốt.

Chương 2

Giới thiệu Học tăng cường

2.1 Khái quát về học tăng cường

Học tăng cường là một nhánh quan trọng trong lĩnh vực học máy, với mục tiêu chính là phát triển các mô hình có khả năng tự động đưa ra các quyết định tối ưu trong môi trường thay đổi liên tục. Thay vì dựa vào dữ liệu huấn luyện có gán nhãn như trong học có giám sát, học tăng cường hoạt động dựa trên nguyên tắc thử nghiệm và học hỏi từ chính các hành động của mình.

Trong học tăng cường, một tác nhân được đặt trong một môi trường cụ thể, và nhiệm vụ của nó là tìm ra chiến lược tối ưu để tối đa hóa phần thưởng tích lũy theo thời gian. Quá trình này không đòi hỏi dữ liệu huấn luyện có sẵn và không yêu cầu việc sửa chữa trực tiếp các hành động chưa tối ưu. Thay vào đó, tác nhân tự học từ những trải nghiệm của mình bằng cách cân bằng giữa việc khám phá các chiến lược mới trong môi trường và khai thác những kinh nghiệm đã tích lũy được.

Học tăng cường tập trung vào việc điều chỉnh hành vi của tác nhân sao cho, trong thời gian dài, nó đạt được phần thưởng cao nhất có thể. Điều này đòi hỏi tác nhân không chỉ phải đưa ra những quyết định đúng đắn trong ngắn hạn mà còn phải tối ưu hóa chiến lược tổng thể để đảm bảo hiệu quả tối đa trong tương lai. [1]

2.2 Mục tiêu bài toán

Trong dự án này, mục tiêu là áp dụng phương pháp học tăng cường để phát triển một mô hình học máy có khả năng chơi trò chơi Snake một cách hiệu quả. Trò chơi Snake, với các yêu cầu phức tạp như tránh va chạm và ăn thức ăn, là một môi trường lý tưởng để thử nghiệm và áp dụng các kỹ thuật học tăng cường. Mục tiêu chính là xây dựng một mô hình có thể tự học và điều chỉnh chiến lược của mình để đạt được điểm số cao nhất trong trò chơi này.

Điều này đòi hỏi tác nhân phải có khả năng học hỏi từ môi trường, tự động điều chỉnh hành vi của mình dựa trên phản hồi nhận được, và đưa ra các quyết định thông minh trong những tình huống phức tạp và thay đổi liên tục.

2.3 Môi trường

Môi trường là yếu tố trung tâm trong học tăng cường, đóng vai trò như một bối cảnh mà tác nhân sẽ tương tác và học hỏi. Trong trò chơi Snake, môi trường bao gồm toàn bộ lưới chơi, vị trí hiện tại của con rắn, vị trí của thức ăn, và các yếu tố cản trở như tường và chính thân rắn. Môi trường sẽ phản hồi lại các hành động của tác nhân bằng cách cung cấp phần thưởng khi tác nhân thực hiện hành động tốt hoặc áp đặt hình phạt khi hành động dẫn đến kết quả không mong muốn.

Trong quá trình chơi, tác nhân phải liên tục đánh giá môi trường xung quanh và đưa ra quyết

định về hướng di chuyển tiếp theo. Môi trường sẽ thay đổi theo từng bước đi của tác nhân, tạo ra các thách thức mới mà tác nhân phải đối mặt và học hỏi để cải thiện chiến lược của mình.

2.4 Tác nhân

Tác nhân trong học tăng cường là thực thể chịu trách nhiệm đưa ra các hành động dựa trên những thông tin nhận được từ môi trường. Trong bài toán này, tác nhân là một mô hình AI được thiết kế để chơi trò chơi Snake. Tác nhân sẽ nhận thông tin về trạng thái hiện tại của môi trường, sau đó đưa ra các quyết định di chuyển, chẳng hạn như di chuyển lên, xuống, trái, hoặc phải.

Mục tiêu của tác nhân là tối đa hóa điểm số bằng cách ăn được nhiều thức ăn nhất có thể, đồng thời tránh va chạm với tường hoặc thân rắn. Tác nhân phải học cách dự đoán các tình huống có thể xảy ra dựa trên những gì đã học từ quá khứ, từ đó đưa ra các hành động giúp đạt được kết quả tốt nhất.

2.5 Phần thưởng

Phần thưởng trong học tăng cường là cơ chế giúp tác nhân đánh giá chất lượng của các hành động mà nó đã thực hiện. Trong trò chơi Snake, phần thưởng thường được trao khi tác nhân ăn được thức ăn, điều này giúp tăng điểm số của trò chơi. Ngược lại, tác nhân sẽ bị phạt khi thực hiện những hành động dẫn đến kết thúc trò chơi, chẳng hạn như va chạm với tường hoặc với thân rắn.

Thiết kế cơ chế phần thưởng phù hợp là rất quan trọng, vì nó sẽ định hướng cho quá trình học hỏi của tác nhân. Một hệ thống phần thưởng tốt sẽ khuyến khích tác nhân thực hiện các hành động mang lại lợi ích lâu dài, giúp cải thiện hiệu suất tổng thể của mô hình.

2.6 Tại sao nên chọn học tăng cường cho bài toán

Học tăng cường là một phương pháp mạnh mẽ và linh hoạt cho các bài toán yêu cầu tác nhân phải đưa ra quyết định trong môi trường không chắc chắn và thay đổi liên tục. Đặc biệt, trong trò chơi Snake, mỗi hành động của tác nhân đều có thể dẫn đến những kết quả khác nhau tùy thuộc vào tình huống cụ thể mà nó đang phải đối mặt. Điều này tạo ra một thách thức lớn trong việc thiết kế chiến lược tối ưu.

Học tăng cường cho phép tác nhân không chỉ tự động tìm ra các chiến lược chơi hiệu quả mà còn có khả năng cải thiện liên tục thông qua việc học hỏi từ môi trường. Bằng cách thử nghiệm các hành động khác nhau và học từ những phản hồi nhận được, tác nhân có thể dần dần tối ưu hóa hành vi của mình để đạt được kết quả mong muốn.

Phương pháp này không chỉ phù hợp với các bài toán như trò chơi Snake mà còn có thể mở rộng để giải quyết nhiều bài toán phức tạp khác trong thực tế, nơi mà việc ra quyết định thông minh và tự động là yếu tố then chốt. Học tăng cường mở ra một hướng đi đầy hứa hẹn cho việc phát triển các hệ thống tự động có hiệu quả.

Chương 3

Xây dựng trò chơi sử dụng Deep Q-Learning

3.1 Thuật toán Deep Q-Learning

Thuật toán Deep Q-Learning (DQN) được triển khai trong dự án này mở rộng từ thuật toán Q-Learning cơ bản, một phương pháp học tăng cường không mô hình tập trung vào việc học giá trị của việc thực hiện một hành động cụ thể trong một trạng thái nhất định. Ở trung tâm của Q-Learning là giá trị Q, được ký hiệu là $Q(s, a)$, đại diện cho phần thưởng tích lũy mà tác nhân có thể đạt được bằng cách thực hiện hành động a trong trạng thái s và sau đó theo đuổi chính sách tối ưu. Giá trị Q được cập nhật bằng cách sử dụng phương trình sau:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \cdot \max_{a'} Q(s', a') - Q(s, a) \right]$$

Ở đây, $Q(s, a)$ phản ánh ước lượng hiện tại của giá trị Q cho cặp trạng thái-hành động (s, a). Thuật ngữ $\delta = [r + \gamma \cdot \max_{a'} Q(s', a') - Q(s, a)]$ đại diện cho sai số khác biệt thời gian (TD error), là sự khác biệt giữa giá trị mục tiêu (ước lượng tốt nhất của giá trị Q thực sự dựa trên phần thưởng trong tương lai) và ước lượng hiện tại $Q(s, a)$. Sai số TD δ đóng vai trò là một thước đo về mức độ cần điều chỉnh ước lượng giá trị Q hiện tại $Q(s, a)$ để phản ánh tốt hơn giá trị thực tế dựa trên trải nghiệm của tác nhân. Khi sai số TD lớn, điều đó cho thấy có sự chênh lệch đáng kể giữa ước lượng hiện tại và giá trị mục tiêu, dẫn đến sự điều chỉnh lớn hơn của giá trị Q. Ngược lại, một sai số TD nhỏ hơn sẽ dẫn đến một cập nhật khiêm tốn hơn. Theo thời gian, khi tác nhân tiếp tục tương tác với môi trường và thu thập nhiều trải nghiệm hơn, sai số TD có xu hướng giảm, khiến giá trị Q ổn định khi chúng hội tụ về chính sách tối ưu. Giá trị Q mục tiêu, $r + \gamma \cdot \max_{a'} Q(s', a')$, kết hợp cả phần thưởng tức thì r và phần thưởng kỳ vọng tối đa từ trạng thái tiếp theo s', xem xét tất cả các hành động có thể a'.

Thuật ngữ $\max_{a'} Q(s', a')$ rất quan trọng vì nó xác định giá trị Q tối đa có thể đạt được từ trạng thái tiếp theo s' bằng cách chọn hành động tốt nhất a' có thể. Giá trị này đại diện cho kết quả tương lai tốt nhất mà tác nhân có thể mong đợi đạt được từ trạng thái s'. Hệ số chiết khấu γ, dao động từ 0 đến 1, xác định tầm quan trọng của các phần thưởng trong tương lai; một giá trị gần 1 khiến tác nhân nhìn xa hơn, trong khi một giá trị gần 0 khiến tác nhân ưu tiên phần thưởng ngay lập tức.

Việc trừ $Q(s, a)$ trong phương trình cập nhật là rất cần thiết để điều chỉnh ước lượng hiện tại của giá trị Q. Sai số TD δ hướng dẫn quá trình điều chỉnh: nếu giá trị Q mục tiêu cao hơn ước lượng hiện tại, việc cập nhật sẽ tăng $Q(s, a)$, và nếu nó thấp hơn, việc cập nhật sẽ giảm $Q(s, a)$. Sự điều chỉnh này đảm bảo rằng các giá trị Q dần dần hội tụ về chính sách tối ưu khi tác nhân tiếp tục tương tác với môi trường và học hỏi từ các trải nghiệm của mình.

Trong bối cảnh Deep Q-Learning, thay vì lưu trữ các giá trị Q trong một bảng, một mạng

nơ-ron được sử dụng để ước lượng các giá trị này, cho phép thuật toán xử lý các không gian trạng thái phức tạp và nhiều chiều [2]. Thuật toán DQN tận dụng một bộ đệm bộ nhớ để lưu trữ các trải nghiệm trong quá khứ, từ đó các mini-batch [3] được lấy mẫu trong quá trình huấn luyện để cập nhật mạng nơ-ron. Cách tiếp cận này, kết hợp với chiến lược epsilon-greedy [4], nơi tác nhân dần chuyển từ việc khám phá sang khai thác, giúp tác nhân học chính sách tối ưu theo thời gian trong khi cân bằng sự đánh đổi giữa phần thưởng ngắn hạn và dài hạn.

3.2 Cấu trúc mạng Neuron

Cấu trúc mạng nơ-ron trong thuật toán Deep Q-Learning (DQN) đóng vai trò quan trọng trong việc xấp xỉ các giá trị Q, đại diện cho phần thưởng tích lũy dự kiến khi thực hiện các hành động cụ thể trong các trạng thái nhất định. Mạng nơ-ron được triển khai trong dự án này là một mạng lưới thần kinh chuyển tiếp được kết nối đầy đủ, được tối ưu hóa để xử lý sự phức tạp của môi trường trong Snake Game. [5]

Tầng Đầu Vào

Tầng đầu vào của mạng bao gồm 12 neuron, tương ứng với biểu diễn trạng thái của trò chơi. Trạng thái này bao gồm các thông tin quan trọng như hướng di chuyển hiện tại của rắn, vị trí tương đối của đầu rắn và thức ăn, và các mối nguy hiểm tiềm tàng từ va chạm trong bước di chuyển tiếp theo. Những đặc điểm này được mã hóa thành một vector 12 chiều $\mathbf{x} = [x_1, x_2, \ldots, x_{12}]$, đóng vai trò là đầu vào cho mạng nơ-ron.

Tầng Ẩn

Mạng nơ-ron có một tầng ẩn với 256 neuron. Mỗi neuron trong tầng ẩn thực hiện một phép biến đổi tuyến tính trên đầu vào, tiếp theo là một hàm kích hoạt phi tuyến sử dụng ReLU (Rectified Linear Unit). Đối với mỗi neuron ẩn h_j, phép tính được thực hiện như sau:

$$h_j = \text{ReLU}\left(\sum_{i=1}^{12} w_{ji}x_i + b_j\right)$$

Điều này có thể biểu diễn cho toàn bộ tầng ẩn như sau:

$$\mathbf{h} = \text{ReLU}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$

trong đó: - $\mathbf{W}_1$ là ma trận trọng số kết nối tầng đầu vào với tầng ẩn. - $\mathbf{b}_1$ là vector bias cho tầng ẩn. - $\mathbf{h}$ là vector đầu ra của tầng ẩn.

Tầng Đầu Ra

Tầng đầu ra bao gồm 3 neuron, mỗi neuron đại diện cho một trong các hành động có thể thực hiện của rắn: đi thẳng, rẽ phải hoặc rẽ trái. Nhiệm vụ của mạng là dự đoán các giá trị Q cho những hành động này dựa trên trạng thái hiện tại làm đầu vào. Giá trị $Q(s, a_k)$ cho mỗi hành động a_k được tính toán như sau:

$$Q(s, a_k) = \sum_{j=1}^{256} w_{kj}h_j + b_k$$

Điều này có thể được tóm tắt cho toàn bộ tầng đầu ra như sau:

$$\mathbf{Q} = \mathbf{W}_2\mathbf{h} + \mathbf{b}_2$$

Trong đó: $\mathbf{W}_2$ là ma trận trọng số kết nối tầng ẩn với tầng đầu ra. $\mathbf{b}_2$ là vector bias cho tầng đầu ra. $\mathbf{Q}$ là vector các giá trị Q cho tất cả các hành động có thể.

Hàm Mất Mát và Tối Ưu Hóa

Mạng nơ-ron được huấn luyện bằng cách sử dụng hàm mất mát Mean Squared Error (MSE), đo lường sự khác biệt giữa các giá trị Q dự đoán $Q(s, a)$ và các giá trị Q mục tiêu $Q_{\text{target}}(s, a)$. Các giá trị Q mục tiêu được tính toán bằng phương trình Bellman:

$$Q_{\text{target}}(s, a) = r + \gamma \cdot \max_{a'} Q(s', a')$$

trong đó: - r là phần thưởng tức thì. - γ là hệ số chiết khấu. - $\max_{a'} Q(s', a')$ là giá trị Q tối đa cho trạng thái tiếp theo s', xem xét tất cả các hành động có thể a'.

Hàm mất mát sau đó được tính toán như sau:

$$\text{Loss} = \frac{1}{N} \sum_{i=1}^{N} [Q_{\text{target}}(s_i, a_i) - Q(s_i, a_i)]^2$$

trong đó N là số lượng mẫu trong mini-batch.

Các tham số của mạng $\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2$ được cập nhật thông qua gradient descent, sử dụng bộ tối ưu Adam:

$$\theta \leftarrow \theta - \alpha \cdot \nabla_\theta \text{Loss}$$

trong đó θ đại diện cho các tham số của mạng, α là tốc độ học, và $\nabla_\theta \text{Loss}$ là gradient của hàm mất mát đối với các tham số.

Huấn Luyện Mạng

Trong quá trình huấn luyện, mạng nhận được các batch trải nghiệm từ bộ nhớ phát lại, trong đó mỗi trải nghiệm bao gồm trạng thái hiện tại, hành động đã thực hiện, phần thưởng nhận được, trạng thái tiếp theo, và liệu tập phim đã kết thúc hay chưa. Mạng học bằng cách giảm thiểu MSE giữa các dự đoán của nó và các giá trị Q mục tiêu được rút ra từ những trải nghiệm này. Theo thời gian, trọng số của mạng được điều chỉnh để tạo ra các dự đoán giá trị Q chính xác hơn, giúp tác nhân đưa ra các quyết định tốt hơn khi chơi trò chơi.

3.3 Tối ưu hóa thuật toán

Để tối ưu hóa hiệu suất của AI Snake, nhiều chiến lược đã được triển khai nhằm cải thiện hiệu quả học tập và sự ổn định của thuật toán Deep Q-Learning. Một trong những phương pháp cốt lõi được sử dụng là điều chỉnh tốc độ học α trong suốt quá trình huấn luyện, kết hợp với việc điều chỉnh kích thước batch để quản lý hiệu quả động lực học của tác nhân.

Tốc độ học thích ứng

Tốc độ học α là một siêu tham số quan trọng trong quá trình huấn luyện, quyết định mức độ điều chỉnh trọng số của mô hình với gradient của hàm mất mát trong mỗi lần cập nhật. Thay vì sử dụng tốc độ học cố định, một chiến lược **tốc độ học thích ứng** đã được áp dụng để tinh chỉnh quá trình học tập dựa trên hiệu suất của tác nhân và sự tiến triển của quá trình huấn luyện.

Chiến lược tốc độ học thích ứng bao gồm việc giảm dần tốc độ học khi quá trình huấn luyện tiến triển. Cách tiếp cận này cho phép tác nhân thực hiện các cập nhật lớn trong giai đoạn đầu của quá trình huấn luyện khi nó đang học các chiến lược cơ bản và cần khám phá không gian giải pháp rộng. Khi hiệu suất của tác nhân cải thiện và bắt đầu hội tụ về chính sách tối ưu, tốc độ học được giảm để cho phép điều chỉnh tinh tế hơn, ngăn chặn mô hình vượt qua giải pháp tối ưu. Các bước sau đây mô tả cách thức tốc độ học thích ứng được triển khai:

1. **Tốc độ học ban đầu**: Quá trình huấn luyện bắt đầu với tốc độ học tương đối cao, cho phép tác nhân nhanh chóng thích nghi với môi trường và học các chiến lược cơ bản. Ví dụ, tốc độ học ban đầu có thể được đặt là $\alpha_0 = 0.001$.

2. **Giảm tốc độ học**: Tốc độ học được giảm tại các khoảng thời gian xác định trước hoặc dựa trên các chỉ số hiệu suất của tác nhân (ví dụ: khi phần thưởng trung bình của tác nhân đạt đến điểm bão hòa). Một phương pháp phổ biến là giảm một nửa tốc độ học sau một số lượng tập nhất định hoặc khi đạt đến ngưỡng hiệu suất cụ thể. Về mặt toán học, tốc độ học sau n tập có thể được cập nhật như sau:

$$\alpha_n = \alpha_0 \cdot \gamma^{\frac{n}{N}}$$

trong đó γ là hệ số suy giảm (ví dụ: 0.95), n là số tập hiện tại, và N là số lượng tập sau đó tốc độ học được giảm một nửa.

3. **Điều chỉnh dựa trên hiệu suất**: Tốc độ học cũng được điều chỉnh dựa trên hiệu suất của tác nhân. Nếu hiệu suất của tác nhân (ví dụ: điểm số trung bình) không cải thiện đáng kể sau một số lượng tập nhất định, tốc độ học sẽ được giảm để cho phép điều chỉnh tinh tế hơn. Điều này giúp ngăn chặn tác nhân bị mắc kẹt trong cực trị địa phương.

Điều chỉnh kích thước batch

Cùng với tốc độ học thích ứng, kích thước batch cho trải nghiệm replay cũng được điều chỉnh động. Kích thước batch được tăng dần thêm 100,000 đến 400,000 mẫu sau mỗi 400 lần lặp lại. Sự tăng này cho phép tác nhân học từ một tập hợp trải nghiệm đa dạng hơn, cải thiện khả năng tổng quát hóa và giảm thiểu overfitting.

Chiến lược kết hợp

Sự kết hợp giữa tốc độ học thích ứng với điều chỉnh kích thước batch tạo ra một chế độ huấn luyện cân bằng giữa việc khám phá và khai thác. Trong giai đoạn đầu của quá trình huấn luyện, tác nhân được hưởng lợi từ các cập nhật lớn hơn và khám phá rộng hơn không gian giải pháp. Khi quá trình huấn luyện tiến triển, tốc độ học được giảm và tác nhân tập trung vào việc tinh chỉnh chính sách dựa trên một loạt trải nghiệm rộng hơn. Chiến lược này tăng cường khả năng của tác nhân trong việc hội tụ đến một chính sách ổn định và hiệu quả đồng thời giảm thiểu nguy cơ dao động hoặc phân kỳ trong quá trình huấn luyện.

3.4 Xây dựng trò chơi Snake AI

3.4.1 Xây dựng trò chơi Snake AI

Việc phát triển trò chơi Snake AI tập trung vào việc triển khai thuật toán Deep Q-Learning (DQN), cho phép tác nhân AI học các chiến lược tối ưu để chơi trò chơi thông qua các tương tác với môi trường. Môi trường trò chơi được xây dựng bằng Pygame, trong khi mô hình Q-Learning được triển khai bằng PyTorch. Phần này trình bày tổng quan về các thành phần chính và các bước liên quan đến việc xây dựng và huấn luyện Snake AI.

Ý tưởng Cốt lõi

Nền tảng của trò chơi Snake AI được xây dựng trên khái niệm học tăng cường, trong đó một tác nhân học cách ra quyết định bằng cách tương tác với môi trường để tối đa hóa phần thưởng tích lũy. Trò chơi Snake là một môi trường lý tưởng cho loại học này do động lực học tương đối đơn giản nhưng thách thức của nó, nơi tác nhân phải điều hướng con rắn để ăn thức ăn trong khi tránh va chạm với tường hoặc chính nó.

Trọng tâm của quá trình này là thuật toán Q-Learning, cho phép tác nhân học giá trị của các hành động khác nhau trong các trạng thái khác nhau của trò chơi. Theo thời gian, tác nhân cải thiện hiệu suất của mình bằng cách tối ưu hóa chính sách của nó, xác định hành động tốt nhất cần thực hiện trong bất kỳ trạng thái nào để tối đa hóa điểm số của nó.

Chiến lược và Thuật toán

Trò chơi AI Snake sử dụng thuật toán Deep Q-Learning (DQN), một sự cải tiến của phương pháp Q-Learning truyền thống, trong đó kết hợp một mạng nơ-ron để xấp xỉ các giá trị Q. Chiến lược huấn luyện AI bao gồm một số bước chính:

- **Khởi tạo**: Mạng Q (mô hình) được khởi tạo với các trọng số ngẫu nhiên. Mô hình này sẽ được huấn luyện để dự đoán các giá trị Q cho các hành động khác nhau dựa trên trạng thái hiện tại của trò chơi.

- **Chọn Hành động**: Tác nhân chọn hành động dựa trên trạng thái hiện tại, hoặc bằng cách khai thác các giá trị Q đã học (chọn hành động có giá trị Q cao nhất) hoặc khám phá các hành động mới (chọn ngẫu nhiên). Sự cân bằng này được quản lý bằng chiến lược epsilon-greedy, trong đó tác nhân khám phá nhiều hơn ở giai đoạn đầu và khai thác nhiều hơn khi đã học.

- **Tương tác với Môi trường**: Tác nhân tương tác với môi trường trò chơi bằng cách thực hiện các hành động. Môi trường phản hồi bằng cách cung cấp phần thưởng hoặc hình phạt dựa trên kết quả của hành động (ví dụ: ăn thức ăn, va vào tường), được sử dụng để hướng dẫn quá trình học tập.

- **Học và Cập nhật Mô hình**: Sau mỗi hành động, tác nhân cập nhật kiến thức của mình bằng cách điều chỉnh các giá trị Q sử dụng phương trình Bellman, kết hợp phần thưởng tức thì và phần thưởng tương lai ước tính. Mạng nơ-ron được huấn luyện bằng cách sử dụng phương pháp phát lại trải nghiệm, trong đó tác nhân lấy mẫu các trải nghiệm từ bộ nhớ để cập nhật mạng Q. Phương pháp này giúp ổn định quá trình huấn luyện và cải thiện hiệu quả học tập.

3.4.2 Cấu trúc Tệp

Dự án được tổ chức thành một số tệp chính, mỗi tệp phục vụ một mục đích cụ thể trong việc phát triển và thực thi AI Snake:

`snake.py`: Tệp này chứa việc triển khai môi trường trò chơi bằng Pygame. Nó bao gồm logic để hiển thị rắn, thức ăn và các thành phần hình ảnh khác, cũng như xử lý các chuyển động của rắn và phát hiện va chạm. Tệp này rất cần thiết vì nó xác định môi trường mà tác nhân tương tác và học hỏi.

`agent.py`: Tác nhân là nhân vật trung tâm trong quá trình học tập, và tệp này xác định hành vi của nó. Nó bao gồm các chức năng để biểu diễn trạng thái, chọn hành động, lưu trữ bộ nhớ và vòng lặp huấn luyện chính. Tệp `agent.py` kết hợp môi trường và mô hình, quản lý sự tương tác giữa chúng.

- *Trình diễn Trạng thái*: Một trong những chức năng quan trọng trong 'agent.py' là hàm biểu diễn trạng thái. Trạng thái là mô tả ngắn gọn về tình huống hiện tại trong trò chơi, chẳng hạn như vị trí của rắn, hướng di chuyển của nó, vị trí của thức ăn và các nguy hiểm tiềm tàng (như tường hoặc thân rắn). Trạng thái này được mã hóa thành định dạng mà mạng nơ-ron có thể xử lý, thường là dưới dạng một vector đặc trưng. - *Chọn Hành động*: Tác nhân chọn hành động dựa trên trạng thái hiện tại. Trọng tâm của quá trình ra quyết định này là chiến lược epsilon-greedy:

- *Khai thác*: Với xác suất $1 - \epsilon$, tác nhân khai thác các giá trị Q đã học bằng cách chọn hành động có phần thưởng dự đoán cao nhất (giá trị Q cao nhất).

- *Khám phá*: Với xác suất ϵ, tác nhân khám phá môi trường bằng cách chọn một hành động ngẫu nhiên. Khám phá này là rất quan trọng trong giai đoạn đầu của quá trình huấn luyện để khám phá các chiến lược hiệu quả.

- *Bộ nhớ phát lại trải nghiệm*: Tác nhân lưu trữ các trải nghiệm của mình trong một bộ đệm phát lại, một cấu trúc dữ liệu lưu trữ các trạng thái, hành động, phần thưởng và trạng thái mới

trong quá khứ. Bộ nhớ này cho phép tác nhân học từ các hành động trước đó bằng cách lấy mẫu các mini-batch trải nghiệm trong quá trình huấn luyện, phá vỡ mối tương quan giữa các trải nghiệm liên tiếp và cải thiện sự ổn định của quá trình học tập.

- *Vòng lặp Huấn luyện*: Vòng lặp huấn luyện là trung tâm của quá trình học tập của tác nhân. Trong mỗi lần lặp lại, tác nhân: - Truy xuất trạng thái hiện tại. - Quyết định một hành động bằng chính sách epsilon-greedy. - Thực hiện hành động trong môi trường trò chơi. - Nhận phần thưởng và quan sát trạng thái mới. - Lưu trữ trải nghiệm vào bộ nhớ phát lại. - Lấy mẫu một mini-batch trải nghiệm từ bộ nhớ. - Sử dụng các trải nghiệm đã lấy mẫu để cập nhật mạng Q bằng cách giảm thiểu hàm mất mát.

`model.py`: Tệp này xác định kiến trúc và chức năng của mạng nơ-ron được sử dụng trong thuật toán Deep Q-Learning. Mạng này rất quan trọng để xấp xỉ các giá trị Q, đại diện cho phần thưởng tương lai dự kiến khi thực hiện các hành động cụ thể trong các trạng thái nhất định. Tệp này mô tả cấu trúc của mạng nơ-ron, bao gồm:

- *Tầng Đầu vào*: Nhận biểu diễn trạng thái (một vector đặc trưng). Số lượng neuron đầu vào tương ứng với số chiều của vector trạng thái.

- *Các Tầng Ẩn*: Các tầng này chịu trách nhiệm học các mẫu phức tạp liên kết trạng thái với các giá trị Q. Số lượng tầng ẩn và số lượng neuron trong mỗi tầng là các siêu tham số có thể được điều chỉnh để tối ưu hóa hiệu suất. ReLU (Rectified Linear Unit) thường được sử dụng làm hàm kích hoạt trong các tầng này để giới thiệu tính phi tuyến và giúp mạng học các hàm phức tạp.

- *Tầng Đầu ra*: Tầng đầu ra có số neuron bằng số lượng hành động có thể thực hiện. Mỗi neuron đầu ra tương ứng với giá trị Q của một hành động cụ thể trong trạng thái đã cho.

- *Lan truyền tiến*: Hàm forward trong 'model.py' xác định cách dữ liệu đầu vào được truyền qua mạng. Nó áp dụng các trọng số và bias đã học tại mỗi tầng và xử lý dữ liệu đầu vào thông qua các hàm kích hoạt, cuối cùng tạo ra các giá trị Q cho mỗi hành động.

- *Hàm Mất mát và Tối ưu hóa*: Mô hình được huấn luyện để giảm thiểu lỗi bình phương trung bình (MSE) giữa các giá trị Q dự đoán và các giá trị Q mục tiêu. Tệp 'model.py' bao gồm logic cho tối ưu hóa này, thường sử dụng bộ tối ưu Adam, phù hợp để xử lý các gradient thưa và yêu cầu ít điều chỉnh hơn so với Phương pháp Gradient Descent truyền thống.

- *Lưu và Tải Mô hình*: Tệp này cũng bao gồm các hàm để lưu mô hình đã huấn luyện vào ổ đĩa và tải nó sau này. Chức năng này rất quan trọng để bảo tồn tiến trình học tập của tác nhân và cho phép triển khai mô hình hoặc huấn luyện tiếp tục ở giai đoạn sau.

- *Lớp QTrainer*: Thường thì 'model.py' có thể bao gồm một lớp trợ giúp (như 'QTrainer') đóng gói logic huấn luyện, bao gồm tính toán mất mát, lan truyền ngược và cập nhật trọng số. Sự trừu tượng hóa này giúp giữ cho mã huấn luyện được tổ chức và mô đun.

Thư mục Mô hình: Thư mục này được sử dụng để lưu trữ các mô hình đã được huấn luyện. Lưu các mô hình tại các điểm kiểm tra khác nhau cho phép bảo tồn tiến trình huấn luyện, cho phép tiếp tục huấn luyện hoặc đánh giá sau này mà không cần bắt đầu từ đầu. Nó cũng tạo điều kiện so sánh giữa các lần huấn luyện khác nhau.

Kết luận

Sự kết hợp giữa 'agent.py' và 'model.py' tạo thành cốt lõi của trí thông minh của Snake AI. Tệp 'agent.py' xử lý việc ra quyết định, học tập và tương tác với môi trường, trong khi 'model.py' xác định cấu trúc và chức năng của mạng nơ-ron xấp xỉ các giá trị Q. Bằng cách tổ chức dự án thành các tệp mô-đun, mỗi tệp có trách nhiệm rõ ràng, cơ sở mã được giữ cho dễ quản lý và thích ứng, tạo điều kiện cho sự phát triển và thử nghiệm thêm với các chiến lược và kiến trúc AI khác nhau.

Chương 4

Huấn luyện và đánh giá

Trong quá trình phát triển một tác nhân thông minh (agent) trong môi trường trò chơi, việc huấn luyện và đánh giá đóng vai trò cực kỳ quan trọng. Huấn luyện là giai đoạn mà tác nhân học hỏi từ các trải nghiệm của mình thông qua việc tương tác với môi trường, giúp tối ưu hóa các quyết định và hành động của mình. Đánh giá thuật toán là bước cần thiết để xác định hiệu quả và khả năng hoạt động của tác nhân sau khi hoàn thành quá trình huấn luyện.

Mục tiêu của việc đánh giá là đo lường hiệu suất của tác nhân trong việc giải quyết các nhiệm vụ cụ thể và đảm bảo rằng nó có thể áp dụng các kiến thức đã học vào các tình huống mới. Quá trình này không chỉ cung cấp thông tin phản hồi về khả năng của tác nhân mà còn giúp điều chỉnh và cải thiện chiến lược huấn luyện để đạt được kết quả tốt nhất.

4.1 Huấn luyện thuật toán

Quá trình huấn luyện là giai đoạn quan trọng nhất trong việc phát triển một tác nhân thông minh (agent) có khả năng đưa ra quyết định tối ưu trong môi trường trò chơi. Mục tiêu chính của quá trình này là giúp tác nhân học hỏi từ các trải nghiệm của mình, cải thiện dần dần chiến lược (policy) để tối đa hóa phần thưởng theo thời gian.

Vòng lặp huấn luyện là nơi tác nhân liên tục chơi nhiều ván game để cải thiện chính sách của mình thông qua các bước sau:

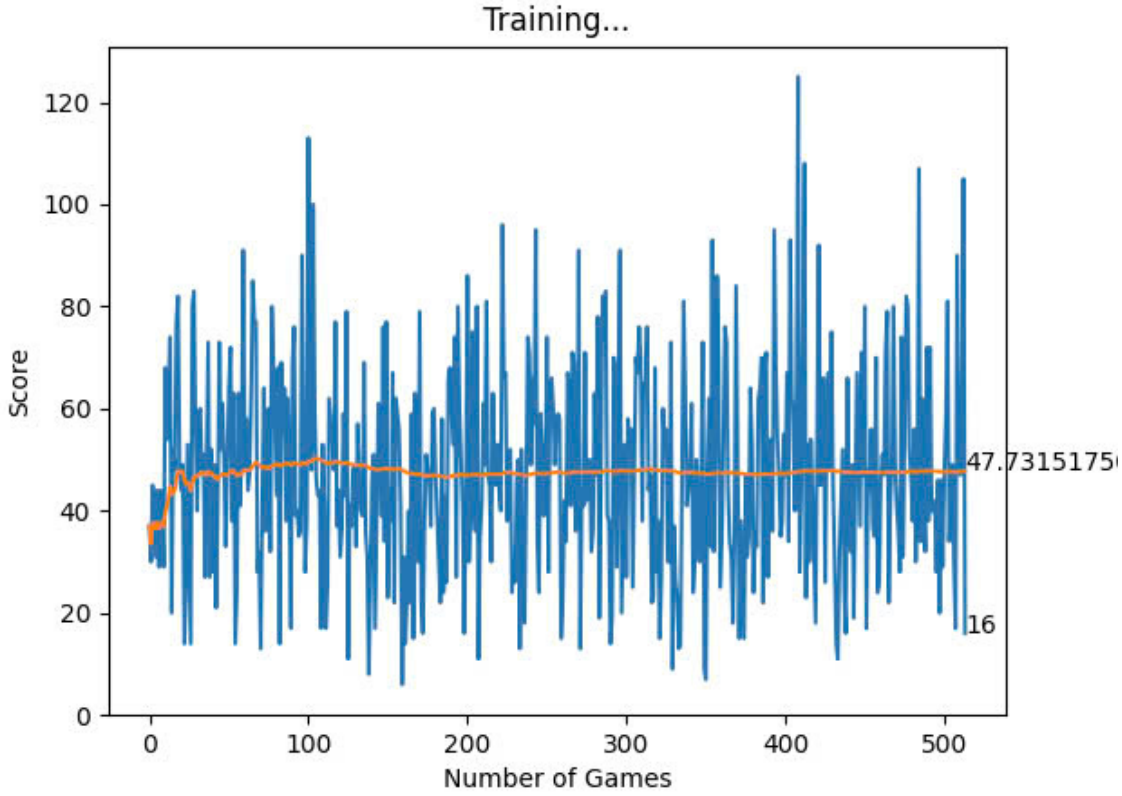
- **Xác định Trạng thái và Hành động:** Tại mỗi bước của trò chơi, tác nhân quan sát trạng thái hiện tại và sử dụng chính sách của mình (dựa trên mạng nơ-ron Q-network) để lựa chọn hành động tối ưu. Hành động này sẽ được thực thi trong môi trường game

- **Nhận Phản hồi và Chuyển Trạng thái:** Sau khi thực hiện hành động, tác nhân nhận được phản hồi từ môi trường dưới dạng phần thưởng, đồng thời trò chơi sẽ chuyển sang trạng thái mới dựa trên kết quả của hành động vừa thực hiện

- **Cập nhật Bộ nhớ:** Trải nghiệm của tác nhân, bao gồm trạng thái hiện tại, hành động đã chọn, phần thưởng nhận được, trạng thái kế tiếp, và trạng thái kết thúc (nếu có), sẽ được lưu vào bộ nhớ để sử dụng trong các bước học sau này

- **Học từ Kinh nghiệm:** Tác nhân sẽ cập nhật mạng nơ-ron Q-network của mình bằng cách học từ các trải nghiệm mới nhất (bộ nhớ ngắn hạn) và định kỳ thực hiện học trên một tập hợp các trải nghiệm trước đó được chọn ngẫu nhiên từ bộ nhớ (bộ nhớ dài hạn). Điều này giúp tác nhân có thể tổng quát hóa tốt hơn và tránh bị quá khớp với các tình huống cụ thể

- **Theo dõi và Đánh giá Hiệu suất:** Trong suốt quá trình huấn luyện, tác nhân liên tục ghi nhận điểm số của mỗi ván game và theo dõi điểm số cao nhất đạt được. Điều này giúp đánh giá sự tiến bộ của tác nhân qua từng ván game và điều chỉnh chiến lược huấn luyện khi cần thiết



Hình 4.1: Quá trình huấn luyện.

4.2 Đánh giá thuật toán

Sau khi hoàn thành quá trình huấn luyện, việc đánh giá hiệu suất của tác nhân trở thành một phần không thể thiếu để đảm bảo rằng tác nhân có thể áp dụng các kỹ năng đã học vào các tình huống thực tế. Đánh giá thuật toán cho phép chúng ta đo lường mức độ thành công của tác nhân trong việc đạt được mục tiêu đã đặt ra.

Thông qua các bài kiểm tra và các chỉ số đánh giá cụ thể, chúng ta có thể xác định được mức độ hiệu quả của các quyết định mà tác nhân đưa ra, từ đó đưa ra những điều chỉnh cần thiết trong quá trình huấn luyện hoặc cải thiện thuật toán. Việc đánh giá này không chỉ giúp xác định khả năng tổng quát của tác nhân mà còn tạo điều kiện cho việc tối ưu hóa các thông số và chiến lược trong tương lai.

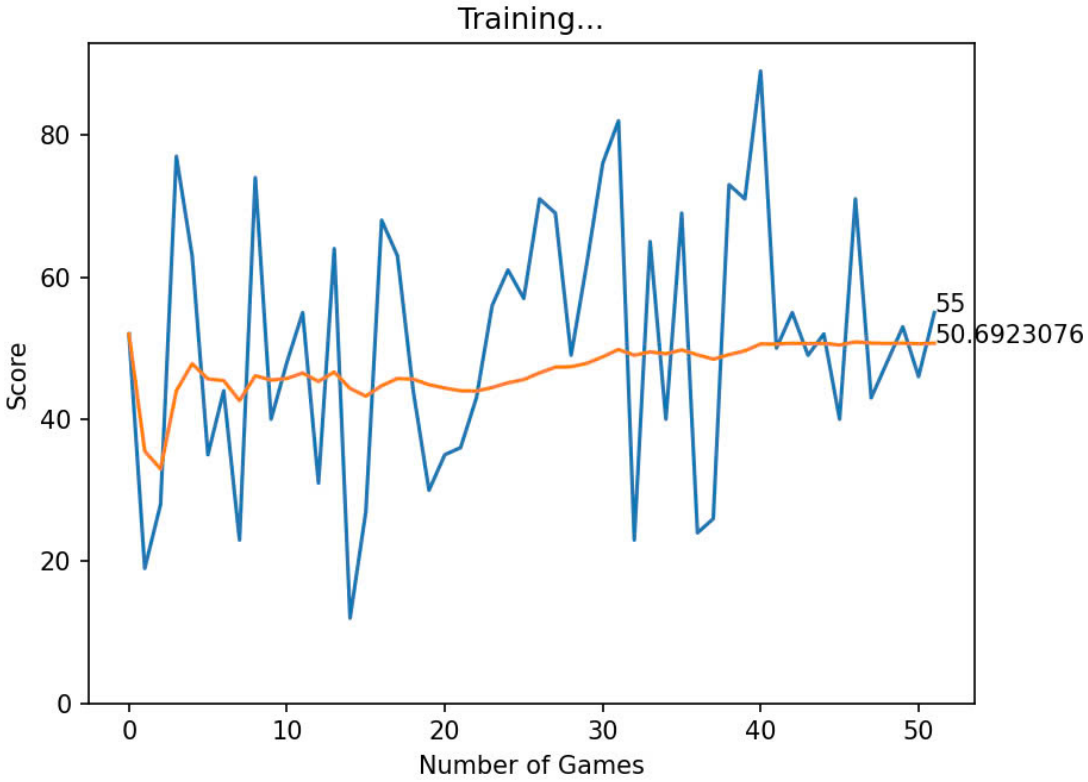
4.2.1. Biến động điểm số

Sau 50 lần chạy, biểu đồ biểu diễn sự thay đổi điểm số (Score) theo số lần chơi (Number of Games). Nhìn vào biểu đồ, ta có thể đưa ra các nhận xét sau:

- Đường màu xanh dương biểu diễn điểm số theo từng lần chơi cho thấy sự biến động đáng kể. Điểm số dao động mạnh từ khoảng 10 đến 80 điểm, thể hiện rằng có sự không ổn định trong quá trình chơi thử.
- Điều này có thể cho thấy sự ngẫu nhiên hoặc tính phức tạp của môi trường huấn luyện.

4.2.2. Xu hướng điểm số trung bình

- Đường màu cam biểu diễn điểm số trung bình theo thời gian. Trong giai đoạn đầu, điểm số trung bình có xu hướng giảm nhẹ, nhưng sau đó duy trì ổn định hơn.



Hình 4.2: Đánh giá mô hình sau 50 lần chơi.

- Điều này cho thấy mặc dù có sự biến động lớn trong điểm số, nhưng trung bình, mô hình dần dần ổn định và có khả năng đạt hiệu suất nhất định.

4.2.3. Điểm số cuối cùng

- Sau lần chạy thứ 50, điểm số đạt được là **55**, trong khi điểm số trung bình là khoảng **50.69**.
- Điều này cho thấy kết quả của mô hình là khá gần với điểm trung bình, chứng tỏ mô hình đã có sự ổn định nhất định sau quá trình huấn luyện.

4.2.4. Cơ hội cải thiện

- Dựa vào xu hướng và biến động của điểm số, có thể mô hình cần được tinh chỉnh thêm về mặt thuật toán hoặc dữ liệu đầu vào để giảm thiểu sự dao động và tăng độ chính xác của mô hình. Việc áp dụng thêm các kỹ thuật như điều chỉnh tham số, sử dụng nhiều dữ liệu huấn luyện hơn hoặc thử nghiệm với các phương pháp huấn luyện khác có thể giúp cải thiện kết quả trong các lần chạy tiếp theo.

Chương 5

Kết quả

Sau quá trình huấn luyện và đánh giá, chúng ta đã thu được những kết quả đáng chú ý, phản ánh khả năng hoạt động của mô hình Deep Q-Learning trong trò chơi Rắn. Các kết quả này cho phép chúng ta đánh giá hiệu suất của tác nhân AI, hiểu rõ hơn về những điểm mạnh và hạn chế của nó, đồng thời cung cấp thông tin quý giá để cải thiện các phương pháp huấn luyện trong tương lai.

5.1 Quá Trình Huấn Luyện

Mô hình đã trải qua quá trình huấn luyện trong 1000 vòng lặp (iterations), trong đó dữ liệu từ môi trường đã được sử dụng để điều chỉnh chiến lược của mô hình. Quá trình huấn luyện được thực hiện với mục tiêu tối ưu hóa việc ra quyết định của tác nhân trong môi trường trò chơi.

5.2 Hiệu Suất Tổng Quát

Trong giai đoạn huấn luyện, mô hình đã thể hiện sự cải thiện về khả năng học hỏi, được thể hiện qua việc điểm số tăng dần theo số lần huấn luyện. Biểu đồ cho thấy sự dao động của điểm số qua các lần huấn luyện, nhưng tổng quan, mô hình đã đạt được mức điểm trung bình khoảng 47.7 sau 1000 lần huấn luyện. Đây là một kết quả đáng khích lệ, thể hiện rằng mô hình đã học được cách tối ưu hóa chiến lược trong một môi trường phức tạp.

5.3 Phân Tích Hiệu Suất

5.3.1 Điểm Số Tốt Nhất

Mô hình đã đạt được điểm số cao nhất là 130 trong một số lần huấn luyện. Điều này cho thấy mô hình có khả năng đưa ra các quyết định gần như tối ưu trong những điều kiện thuận lợi.

5.3.2 Điểm Số Thấp Nhất

Mô hình cũng đã ghi nhận điểm số thấp nhất là 10 trong một số vòng huấn luyện. Sự sụt giảm điểm số này có thể cho thấy mô hình vẫn chưa hoàn toàn ổn định và có thể gặp khó khăn trong việc đưa ra quyết định trong các tình huống phức tạp hoặc chưa từng gặp phải.

5.4 Phân Tích Biến Động

Sự biến động trong hiệu suất huấn luyện của mô hình là đáng kể, với điểm số dao động từ 10 đến 130. Mặc dù có sự dao động, xu hướng chung cho thấy điểm số đang tăng dần theo thời gian,

cho thấy mô hình đang dần học hỏi và cải thiện.

5.5 Kết Quả Cuối Cùng

Sau quá trình huấn luyện, mô hình đã đạt được những kết quả sau:

- Điểm số trung bình qua 500 lần huấn luyện: **47.73**
- Điểm số cao nhất đạt được: **130**
- Điểm số thấp nhất ghi nhận: **10**

Những kết quả này cho thấy mô hình đã phát triển được một số chiến lược hiệu quả, mặc dù vẫn còn những biến động đáng kể trong hiệu suất. Các bước cải tiến tiếp theo có thể bao gồm việc tối ưu hóa thêm các tham số mô hình và tiếp tục huấn luyện để giảm thiểu sự biến động này.

Chương 6

Đề xuất cải thiện

Mô hình học tăng cường hiện tại có khả năng đạt hơn 100 điểm trong trò chơi Snake và có thể chơi với số điểm trung bình là khoảng 50 điểm. Tuy vậy, mô hình này vẫn có nhiều điểm cần được cải thiện.

6.1 Mạng neuron

Mô hình hiện tại chỉ sử dụng một tầng neuron ẩn, điều này có thể hạn chế khả năng học tập và khái quát hóa của mô hình. Một vài đề xuất để cải thiện mạng neuron là:

- **Thêm tầng ẩn**: Bổ sung thêm các tầng neuron ẩn có thể giúp mô hình học được những đặc trưng phức tạp hơn từ môi trường, từ đó đưa ra các quyết định chính xác hơn trong quá trình chơi.
- **Áp dụng kỹ thuật tối ưu hóa**: Sử dụng các phương pháp tối ưu hóa như Learning Rate Scheduling hay thử nghiệm với các hàm kích hoạt phi tuyến tính khác nhau như Sigmoid hay Leaky ReLU để có thể giúp mô hình tránh hiện tượng quá khớp và cải thiện khả năng tổng quát hóa.

6.2 Tối ưu hóa mô hình

Các tham số trong mô hình hiện tại có thể được điều chỉnh và thí nghiệm để có thể tìm ra được các tham số phù hợp nhất và đưa lại hiệu suất cao nhất có thể. Phương pháp tối ưu hóa tham số là:

- **Tinh chỉnh hyperparameter (tham số đặt trước)**: Tinh chỉnh các hyperparameter như tốc độ học và kích thước batch có thể giúp cải thiện hiệu quả học tập của mô hình.
- **Áp dụng kỹ thuật tăng cường dữ liệu (Data Augmentation)**: Tăng hiệu suất của mô hình bằng cách thay đổi môi trường chơi, thay đổi tốc độ trò chơi hoặc thêm các thử thách mới (VD: loại đồ ăn mới) có thể giúp mô hình học tốt hơn và trở nên linh hoạt hơn trong các tình huống khác nhau.

Chương 7

Kết luận

Dự án này không chỉ chứng minh hiệu quả của việc áp dụng phương pháp học sâu Q-learning trong việc đào tạo một tác nhân AI chơi trò chơi Snake mà còn mở ra những khả năng mới cho việc áp dụng các kỹ thuật này trong nhiều lĩnh vực khác.

Kết quả đạt được cho thấy cách mà các phương pháp học tăng cường kết hợp với học sâu có thể giúp các hệ thống AI học hỏi và phát triển các hành vi phức tạp từ dữ liệu thô, từ đó giải quyết được những bài toán thực tiễn một cách hiệu quả.

Dự án cung cấp một nền tảng vững chắc cho các nghiên cứu và ứng dụng tiếp theo trong lĩnh vực AI, cho thấy tiềm năng to lớn của những công nghệ này trong việc mở rộng khả năng của các hệ thống học máy và trí tuệ nhân tạo.

Tài liệu tham khảo

[1] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," MIT Press, 2018.

[2] S. Tandon, "What is a high dimensional state in reinforcement learning?," 2018. Available online: https://ai.stackexchange.com/questions/9812/what-is-a-high-dimensional-state-in-reinforcement-learning.

[3] A. Bilogur, "Full batch, mini-batch, and online learning," 2018. Available online: https://www.kaggle.com/code/residentmario/full-batch-mini-batch-and-online-learning.

[4] S. Shawl, "Epsilon-greedy algorithm in reinforcement learning," 2023. Available online: https://www.geeksforgeeks.org/epsilon-greedy-algorithm-in-reinforcement-learning.

[5] M. S. Ausin, "Neural network in deep q learning," 2020. Available online: https://markelsanz14.medium.com/introduction-to-reinforcement-learning-part-3-q-learning-with-neural-networks-algorithm-dqn-1e22ee928ecd.