Chơi các trò chơi trên hệ máy Atari 2600 bằng mô hình Double Deep Q Learning

TRẦN MINH HIẾU (20170075)

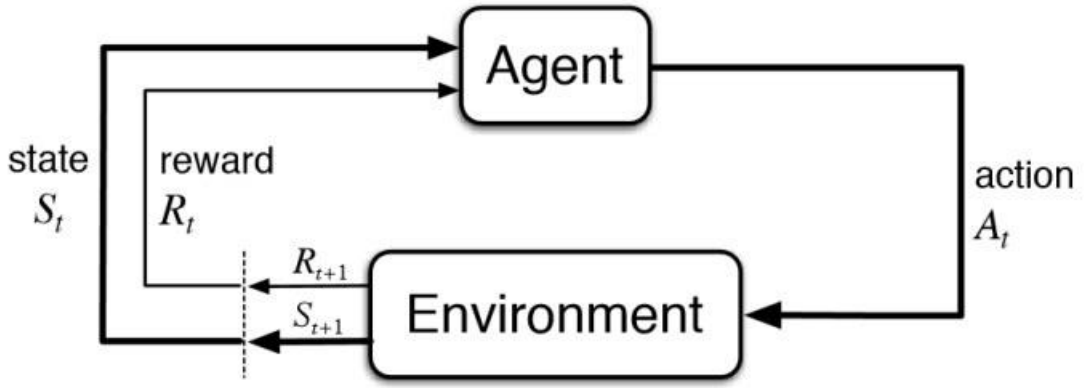
TRƯƠNG QUANG KHÁNH (20170083)

TRƯƠNG NGỌC GIANG (20170067)

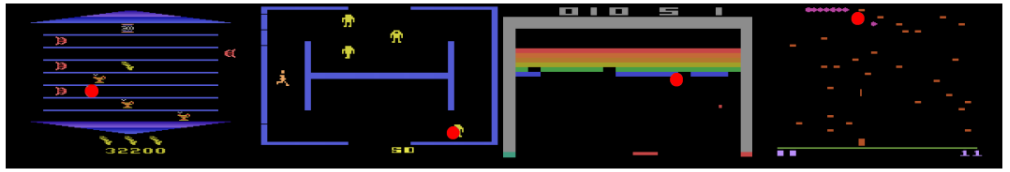
Reinforcement Learning

- Một trong ba mô hình cơ bản của Machine Learning.
- Khác biệt:
  - Không cần dữ liệu gắn nhãn trước.
  - Hướng tới xây dựng mô hình thông minh, có thể đưa ra quyết định.
  - Tối ưu mục tiêu được đặt ra.

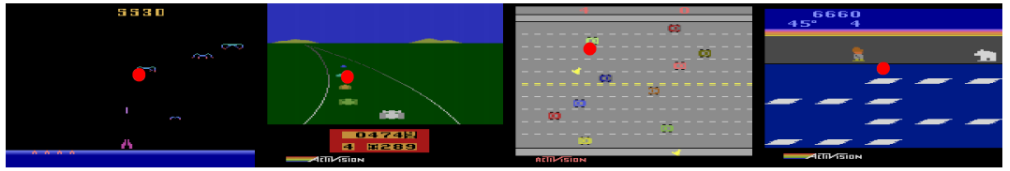
Mô hình Reinforcement Learning



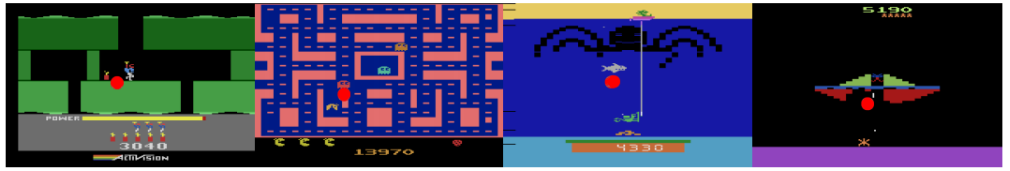
Atari 2600



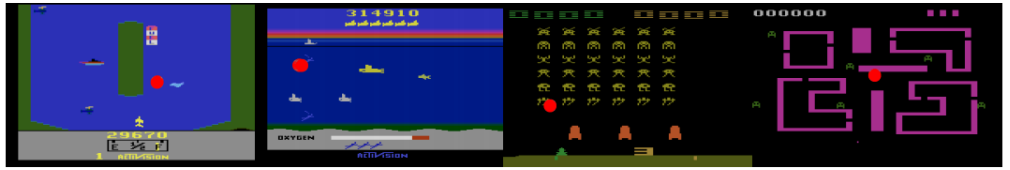
(a) Asterix (b) Berzerk (c) Breakout (d) Centipede



(e) Demon Attack (f) Enduro (g) Freeway (h) Frostbite



(i) Hero (j) Ms.Pacman (k) NTG (l) Phoenix



(m) Riverraid (n) Seaquest (o) Space Invaders (p) Venture

Q Learning

- $Q(s, a)$: giá trị kì vọng của việc thực hiện hành động a tại trạng thái s.
  - Nếu s kết thúc game → giá trị cố định với mọi a (thường là tiêu cực).
  - Nếu s chưa kết thúc game và biết trạng thái kế tiếp s':

$$Q(s, a) = E_{\forall s' possible\ from\ s}[r_a(s, s') + \gamma \max_{\forall a'} Q(s', a')]$$

  - γ: Future discount coefficient – [0.9, 1].
    - $\gamma = 1 \rightarrow Q(s, a) = \sum r_a(s, s')\ \forall\ (s, a, s')\ until\ end\ of\ game$
    - $\gamma < 1 \rightarrow$ Ít quan tâm tới các phần thưởng về sau hơn.

- Trên thực tế không bao giờ tìm được $Q(s, a)$ chính xác, chỉ có thể tìm ước lượng $Q^*(s, a)$.

→ Chúng ta có thể áp dụng Deep Learning vào đây!

Cách tìm $Q^*(s, a)$

Khi model làm tốt

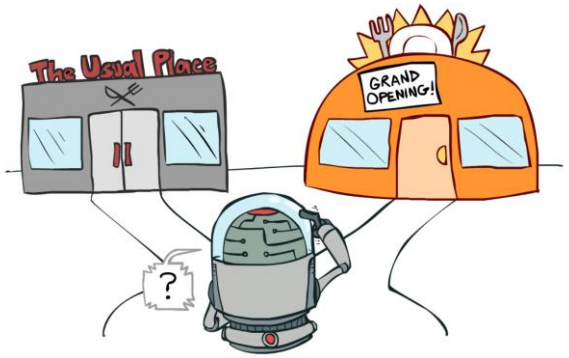
Khi model làm dốt

Tối ưu hàm $Q^*(s, a)$ bằng chính nó

1. Lấy bộ ba kinh nghiệm (s, a, s').
2. Tính $Q^*(s', a')$ $\forall$ a'.
3. Tính $Q^*(s, a)$ dùng công thức ở slide trước.
4. Gradient Descent để tối ưu $Q^*(s, a)$ hiện tại.

Exploration vs Exploitation

- Exploration: Khám phá khả năng có thể của môi trường.
- Exploitation: Lợi dụng kiến thức đã biết để tối ưu mục tiêu.
- Cài đặt: Hệ số ε – xác suất hành động ngẫu nhiên/hành động dự đoán tốt nhất hiện tại.
  - Giảm dần qua thời gian.



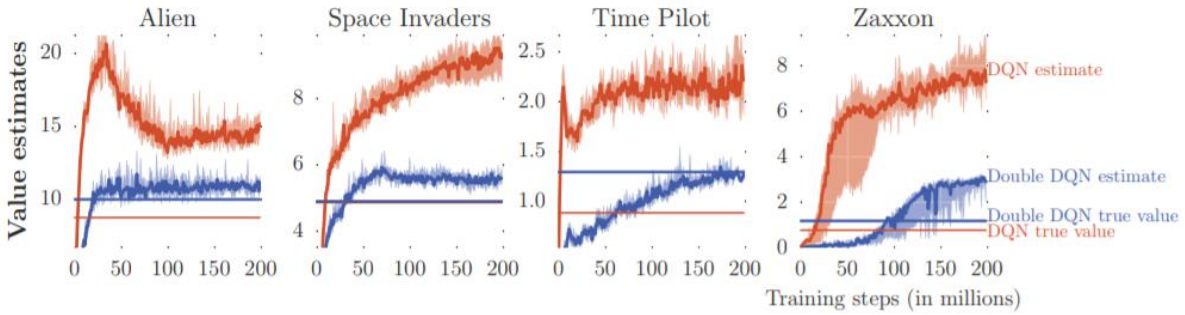
Experience Replay

- Chọn kinh nghiệm gì để luyện tập?
  - Chọn tất cả?
    - Dữ liệu gần nhau quá giống nhau → ít giá trị.
    - Dữ liệu gần nhau chỉ gồm các hành động giống nhau → dữ liệu không cân bằng, kinh nghiệm mới ít được tiếp thu.
  - Chọn ngẫu nhiên một số lượng nhỏ
    - Dữ liệu gần nhau ít khả năng chọn cùng nhau
    - Số lượng nhỏ → Dữ liệu mới có thể gây ảnh hưởng.

Vấn đề overestimation

- Ưu tiên exploit các hành động giá trị cao.
  → Xu hướng giá trị dự đoán bị đẩy cao lên.

- Giả sử các hành động ban đầu đều có dự đoán bằng 0.
  Sau một lần cập nhật, một số lớn hơn 0, một số nhỏ hơn 0.
  → Ít có cơ hội sửa chữa các hành động sai.



Phân tách làm hai mô hình

- Mô hình luyện tập $Q^*(s,a)$, mô hình dự đoán $Q'(s,a)$.
- Khởi tạo với trọng số bằng nhau.
- $Q'(s,a)$ cập nhật lại bằng $Q^*(s,a)$ sau một khoảng thời gian.
- $Q^*(s,a) = E_{\forall s' possible\ from\ s}[r_a(s,s') + \gamma \max_{\forall a'} Q'(s',a')]$

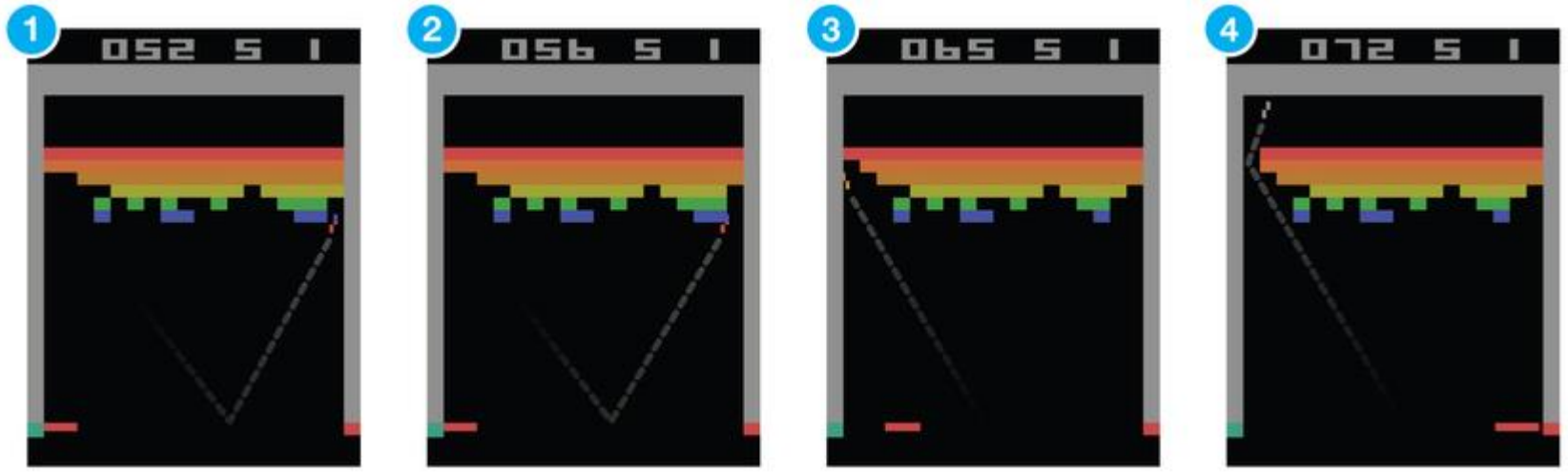
→ Có cơ hội để $Q^*(s,a)$ cập nhật lại, tránh overestimation.

Thuật toán

1. Khởi tạo bộ nhớ kinh nghiệm D với kích cỡ N. Khởi tạo mô hình luyện tập $Q^*(s,a)$ với trọng số ngẫu nhiên, và mô hình dự đoán $Q'(s,a)$ với trọng số bằng $Q^*(s,a)$.
2. For episode $= 1\ to\ M$ do:
    1. Đưa môi trường về trạng thái khởi tạo.
    2. For $t = 1\ to\ T$ do:
        1. Với xác suất ε đưa ra hành động ngẫu nhiên a_t, ngược lại đưa ra hành động $a_t = argmax\ Q^*(s_t, a_t)$.
        2. Nhận lại phần thưởng $r_{a_t}(s_t, s_{t+1})$ và trạng thái mới s_{t+1}.
        3. Lưu kinh nghiệm $\varphi_t = (s_t, a_t, s_{t+1})$ vào bộ nhớ D.
        4. Lấy ngẫu nhiên tập hợp kinh nghiệm $\{\varphi_{i_1}, \varphi_{i_2}, \dots, \varphi_{i_n}\}$ từ D.
        5. Thực hiện tối ưu $Q^*(s,a)$ tới các giá trị $y_i = E_{\forall s' possible\ from\ s}\left[r_a(s,s') + \gamma \max_{\forall a'} Q'(s',a')\right]$.
        6. If shouldUpdateTarget() then $Q'(s,a) = Q^*(s,a)$

Tiền xử lý dữ liệu

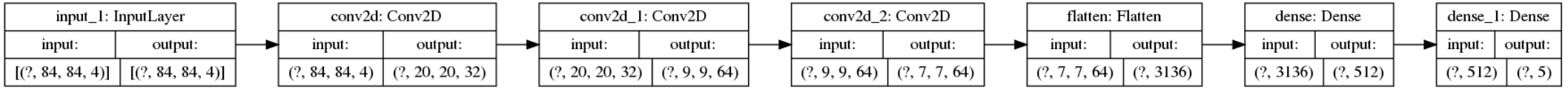
- Resize và cắt hình ảnh giao diện từ 210×160 xuống 84×84.
- Đưa từ RGB về greyscale (ảnh đen trắng).
- Chập 4 khung hình liên tiếp vào làm một.



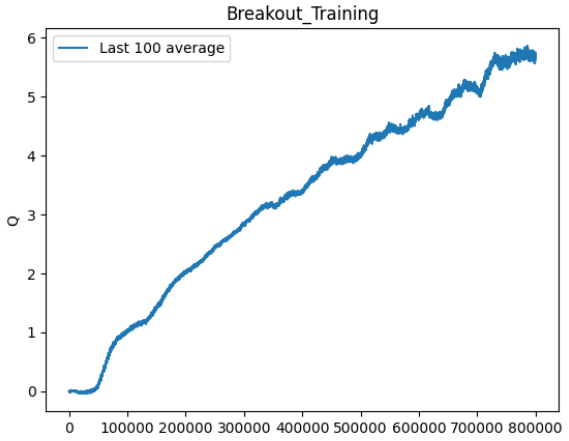
Tiền xử lý dữ liệu (tiếp)

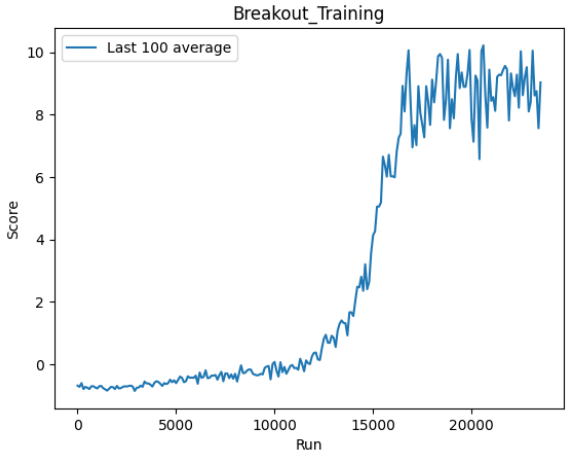
- Tất cả các giá trị $r_{a_t}(s_t, s_{t+1})$ được thay bằng $sign(r_{a_t}(s_t, s_{t+1}))$ có giá trị -1, 0 hoặc 1.
  - Chỉ quan tâm tới hành động tốt hay xấu.
  - Có lợi cho một số game như Breakout – các khối bị phá có giá trị khác nhau.
- Trạng thái kết thúc cho giá trị -1.
  - Hướng tới tránh việc kết thúc game.

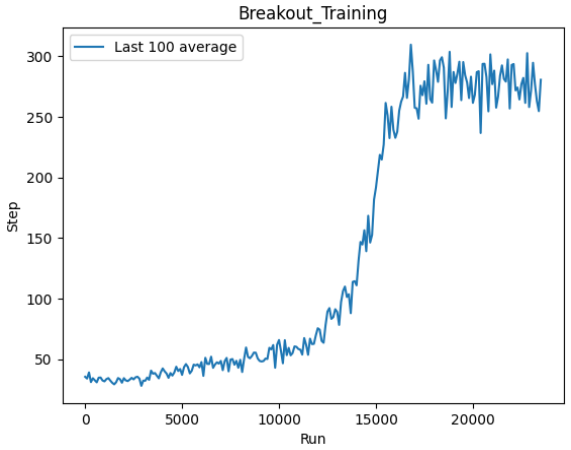
Mô hình Deep Learning của $Q^*(s, a)$



Biểu diễn kết quả







Q&A