

# 1. Mạng Neural và Ước lượng Giá trị Hành động (Action-Value Function)

- Ta dùng một **mạng neural** có một **node đầu ra** cho mỗi hành động.
- Input của mạng là **state** (vị trí, vận tốc của tàu đổ bộ, v.v...).
- Output là **giá trị ước lượng  $Q(s,a)$**  cho từng hành động  $a$ .

## Cập nhật Q:


Sử dụng TD error:

$$\delta = R_{t+1} + \gamma \sum_a \pi(a|S_{t+1})Q(S_{t+1}, a) - Q(S_t, A_t)$$

Chỉ cập nhật weights ở layer cuối cho **action** được chọn.

Chỉ cập nhật weights ở layer cuối cho **action** được chọn.

- Điều này không có vấn đề lớn vì:
  - Các **layer trước (feature layer)** được **chia sẻ** cho mọi action → các lần cập nhật giúp học representation tốt hơn và **tổng quát hơn**.

Cách thiết kế	Ưu điểm	Nhược điểm	
Một mạng chung có nhiều output	Shared representation → tổng quát tốt hơn	Có thể gây xung đột cập nhật	
Một mạng riêng cho mỗi action	Tránh xung đột	Học representation yếu hơn do ít dữ liệu mỗi mạng	

ADAM Optimizer:

- Là sự kết hợp của:
  1. **Vectorized Step Sizes** (như Adagrad/RMSprop)
  2. **Momentum** (như SGD với Momentum)

**Thành phần chính:**

- **Gradient trung bình:** dùng để tính **momentum**.
- **Bình phương gradient trung bình:** dùng để tính **step-size riêng cho từng weight**.
- **Thông số cần quan tâm:**
  - **beta\_m:** kiểm soát momentum.
  - **beta\_v:** kiểm soát step-size vector.
  - **epsilon:** giá trị nhỏ trong mẫu số để tránh chia 0.
  - **alpha:** **global learning rate** (tham số bạn sẽ điều chỉnh trong Capstone).