DQN

Goal: Approximate the optimal action-value function

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \,\middle|\, s_0 = s, a_0 = a, \pi\right]$$

$$Q^*(s,a) = \mathbb{E}\left[r + \gamma \max_{a'} (s', a') \,|\, s, a\right]$$ – Bellman optimality condition

Classical Q-Learning and the Bellman equation

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

Bellman equation

If $\alpha = 1 \rightarrow Q(s,a) \leftarrow r + \gamma \max_{a'} Q(s', a')$

Approximation with a NN

$$Q(s, a; \theta) \approx Q^*(s,a)$$

Stabilize learning: Two-Network Architecture

Challenges: Instability due to non-stationary targets

Policy Network: Estimate Q-values for current state

- Updated continuously at every training step

<u>Target Network</u>: Provides stable target Q-values during the training of the Policy Network.

- Copy of Policy Network but updated less frequently

<u>Implementation:</u>

- <u>Action selection</u>: Policy Network (state)
- <u>Environment interaction</u>: Action executed $\mapsto s', r$
- <u>Experience storage</u>: (s, a, r, s') stored in replay buffer

  Experience replay
- <u>Sampling & Training</u>:
  - * - Sample from replay buffer - Avoids overfitting, improves generalization
  - Compute target Q-value: $y = r + \gamma \max_{a'} Q_{target}(s', a')$
- <u>Updating the Policy Network</u>
  - $L = [y - Q_{policy}(s,a)]^2$ (Loss) → Update with gradient descent (Lectures 1&5)

*

```
q-value = policy-net(shape)      - dimension (batch_size, actions)
                                          A1   A2
                                     B1 [[24, 12]
                                     B2  [13,  1]]

policy-net.gather(1, actions)        actions = [[0],
                                                [0]]
        => [[24],
            [13]]

.squeeze() => [24, 13] = q-values
```

target_net(next_state) $\begin{array}{ccc} & a_1 & a_2 \\ b_1 & [[\,2, & 0\,], \\ b_1 & [\,0 & 3\,]] \end{array}$

~~targetnet~~'.max(1) ↦ ([2,3], [0,1])

[0] ↦ [2,3] = ~~target~~ next_q_value   if dones = True = 1

target_q_value = $r + \gamma \cdot$ next_q_value $\cdot$ (1-dones)

<u>Recall</u>

$$\underbrace{Q(s,a)}_{\text{Policy\_net}} = r + \gamma \max_{a'} \underbrace{Q(s',a')}_{\text{Target\_net}}$$ (Bellman equation

$\Rightarrow$ Loss = |target_q_value − q_value|^2