上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Lecture 32: Deep Q-Networks

Tao Huang

John Hopcroft Center, School of Computer Science, Shanghai Jiao Tong University

https://taohuang.info/cs3317

https://oc.sjtu.edu.cn/courses/89538

AI tools assisted in generating some figures in these slides. All such content has been reviewed, and the instructor is responsible for its accuracy.

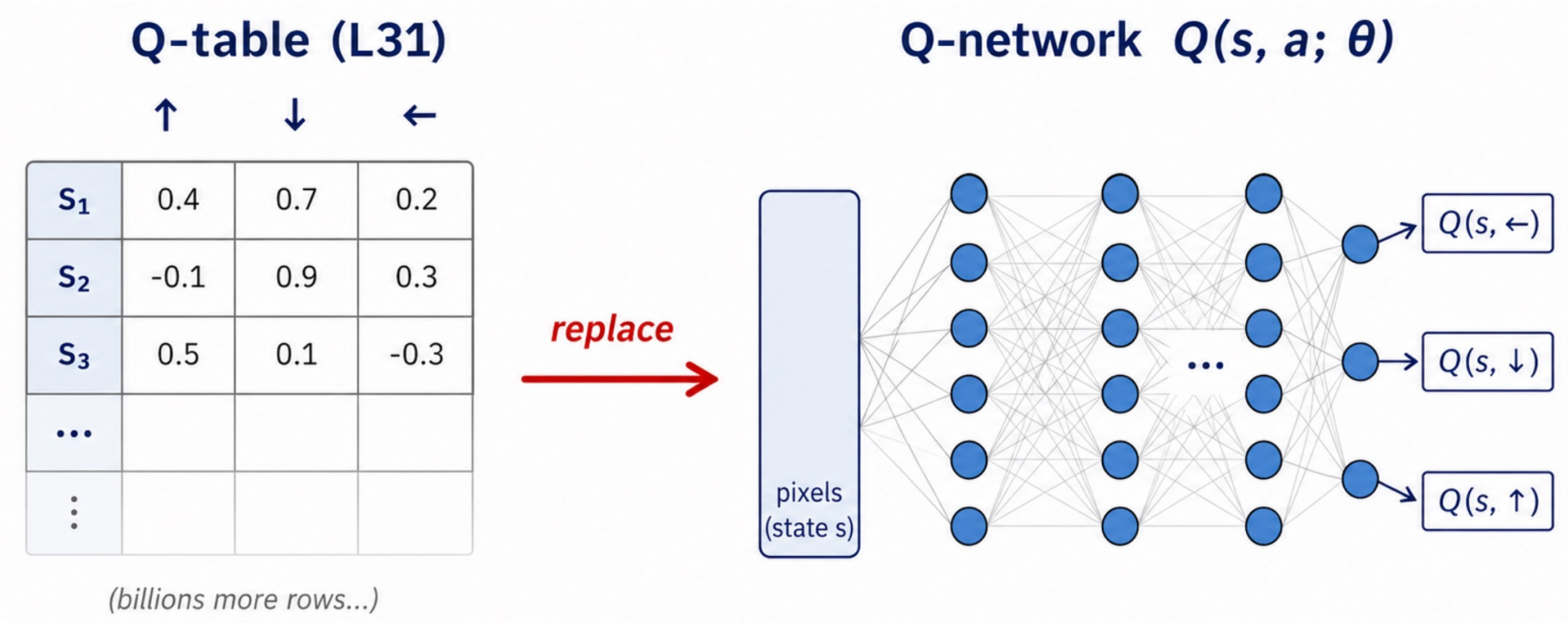
Where We Left Off

- **L29:** MDP formalism.
- **L30:** solve it when P, R are known (DP).
- **L31:** learn from samples (Q-learning) — but keep a Q-table.

Q-table with 10^{250} entries? You don't have that much disk.

The Big Idea

- Replace the Q-table with a neural network: Q(s, a; θ).
  - *Function approximation turns a lookup into a learned function — and lets RL leave gridworld.*



Objectives

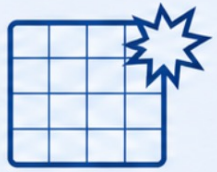
By the end of this lecture you should be able to:

- Explain why a tabular Q-function can't scale to Atari.
- State the DQN loss and identify the regression target.
- Describe what experience replay does and why it helps.
- Describe what the target network does and why it helps.
- Place DQN in the line from Q-learning → AlphaGo.

Outline

1

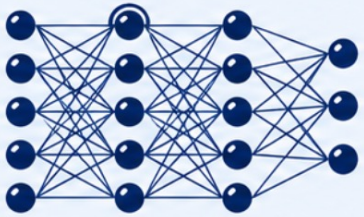
Part 1.
Why tables don't scale



The state-space explosion problem.

2

Part 2.
Deep Q-Networks



Function approximation + two stability tricks.

3

Part 3.
The Atari moment & beyond

Nature 2015 → AlphaGo.

1. Why Tables Don't Scale

Situation: Tabular Q-learning Works

Recall the update from L31:

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

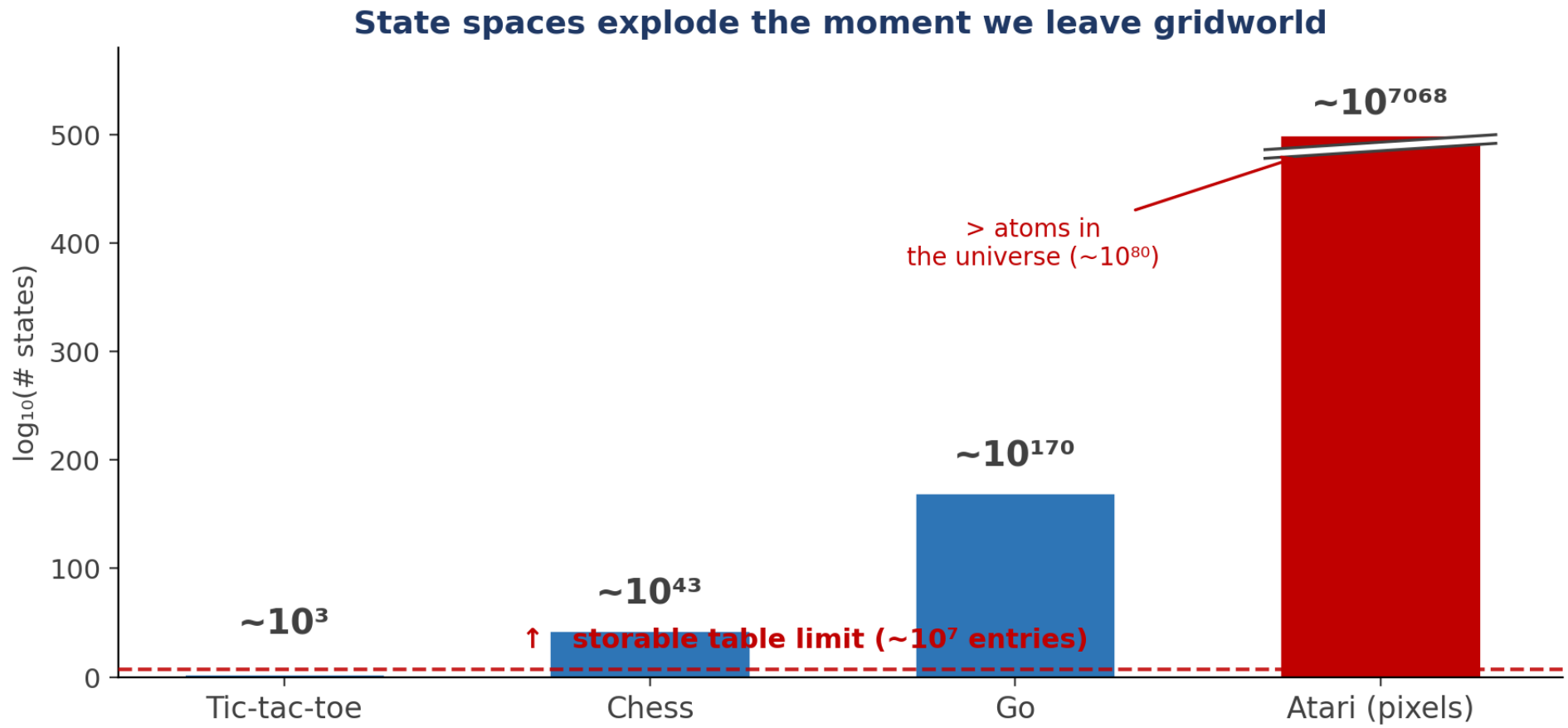
One number per (s, a) cell. Converges.

Situation: L31's tabular Q-learning is correct and convergent.

Complication: It needs |S| × |A| numbers in memory.

Question: *What is* |S| *for the games we actually want to play?*

Complication: States Explode



At 84×84 grayscale frames, the state space alone is larger than the number of atoms in the universe — many times over.

Why You Can't Just Discretize

- **Pixels are high-dim and structured**
  (nearby pixels correlate).

- **Most states are never visited**
  — the table is mostly empty.

- **Even if you did visit them, you wouldn’t generalize.**
  State s and a near-identical s' would get separate cells.

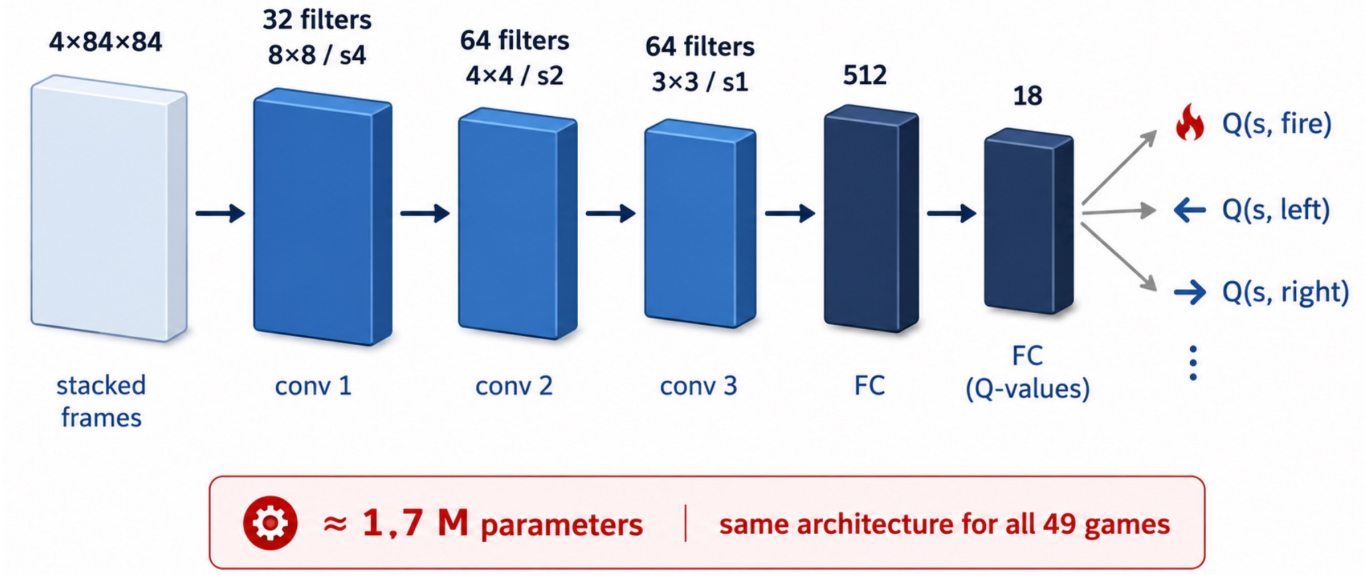
We need a function that says "similar states → similar Q-values" — for free.

Answer: Function Approximation

- Replace $Q(s, a)$ with $Q(s, a; \theta)$ — parameters θ, not entries.
- *For Atari:* θ *is the weights of a small CNN.*

Two properties we get from this:

- **Generalization** — pixels close in input space get similar outputs.
- **Compression** — millions of states, ≈1.7M weights.



2. Deep Q-Networks

Idea: Turn Q-learning into Regression

- Recall L31's TD target: $r + \gamma \max Q(s', a')$. Treat it as the label.
- Train the network to minimize squared error against this label:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta) - Q(s, a; \theta)\right)^2\right]$$

Q-learning with a neural net inside. That's it — almost.

Think: What Could Go Wrong?

We just plugged a neural net into Q-learning. Why didn't this work for 25 years?

Discuss with your neighbor — three prompts:

(a) Consecutive Atari frames are nearly identical. What does that do to SGD?

(b) The label $r + \gamma \max Q(s', a'; \theta)$ uses the same θ being trained. What problem does that create?

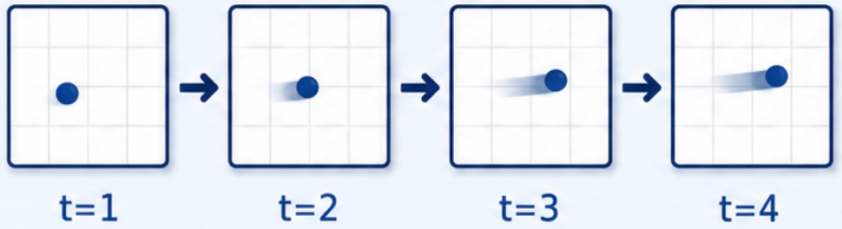
(c) If a tiny weight change shifts every Q-value at once, what happens to the action you pick?

The Two Problems

1 Correlated samples

Consecutive frames are nearly identical.

SGD assumes i.i.d. samples.

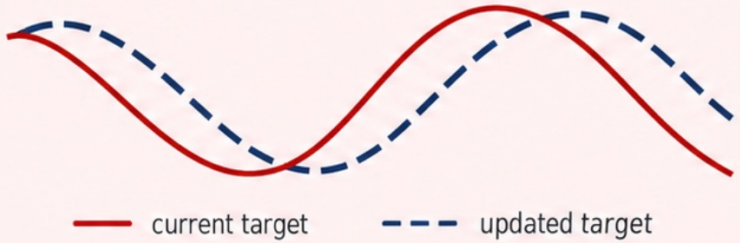


→ *unstable, biased gradients*

2 Moving target

The target $r + \gamma \max Q(s'; \theta)$ uses θ.

Update $\theta \rightarrow$ the target moves too.

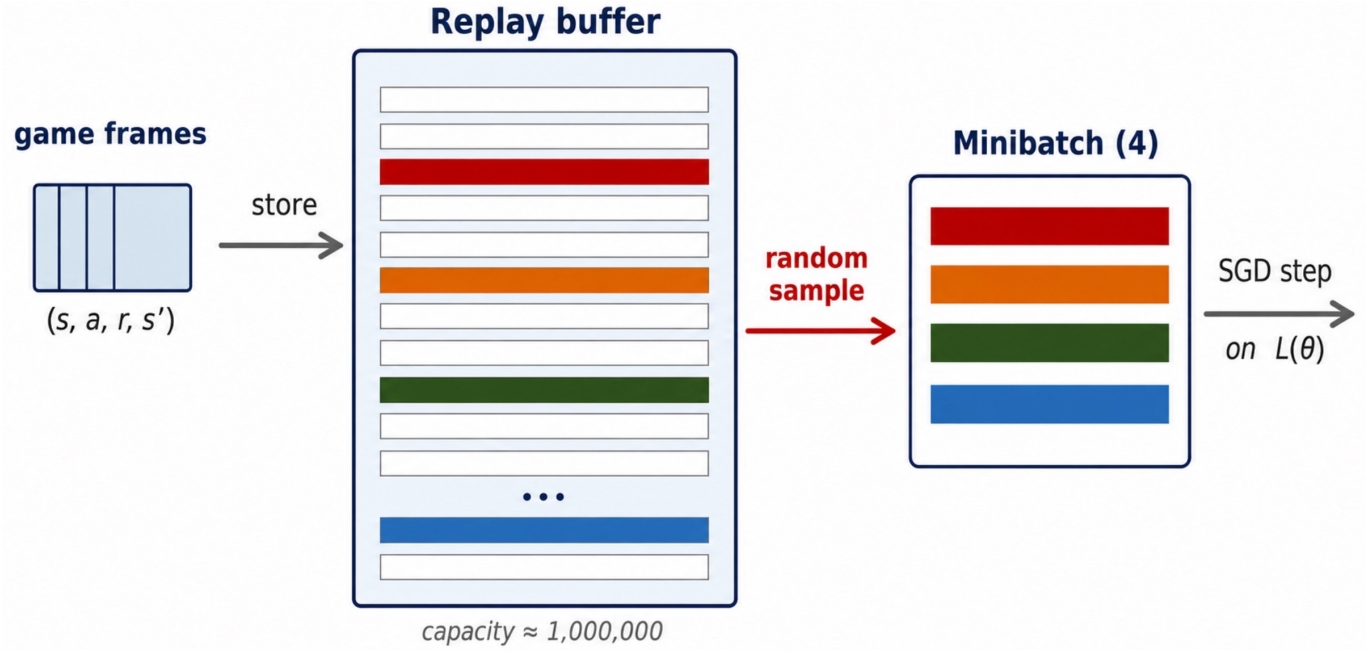


→ *chasing your own tail*

DQN's two ideas are aimed at exactly these two problems.

Trick 1: Experience Replay

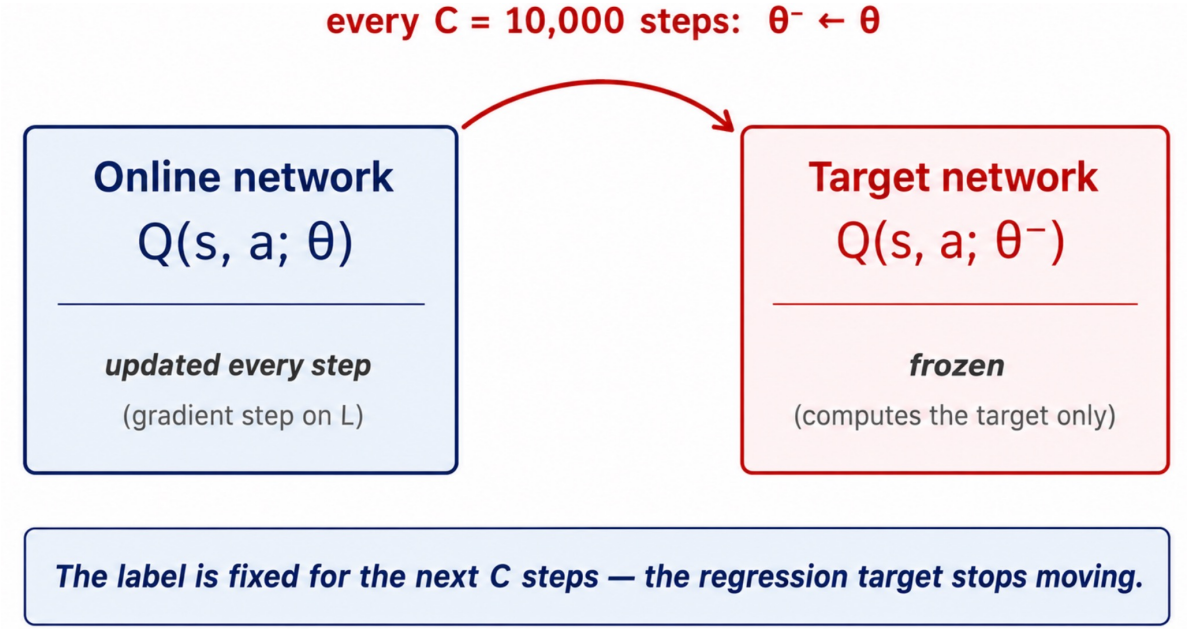
- Store every transition **(s, a, r, s')** in a circular buffer *(capacity ≈ 10^6)*.
- At each training step: sample a **random minibatch** from the buffer.



- **Breaks correlation** (random batch ≈ i.i.d.)
- **Reuses experience** (every transition trains the net many times).

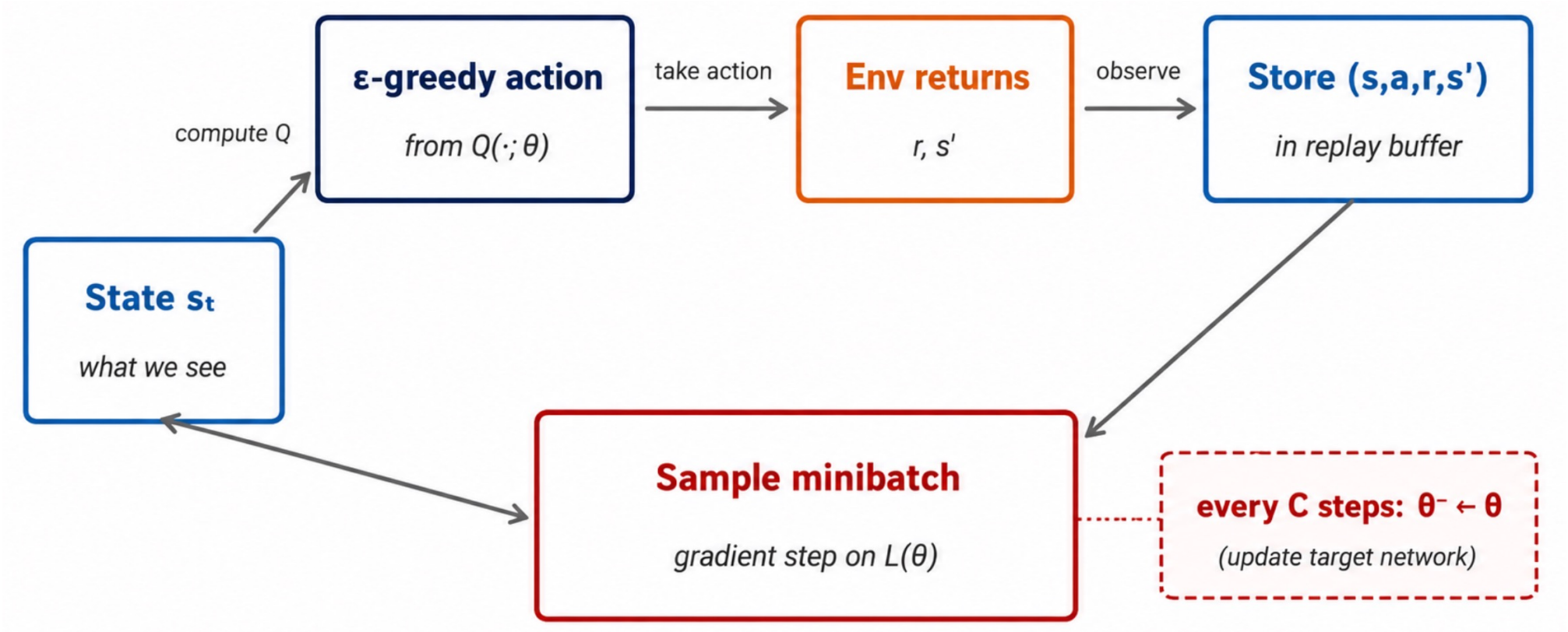
Trick 2: Target Network

- Keep a **second, frozen** copy of the network **$Q(s, a; \theta^{-})$.** Use it to compute the target.
- Every C steps (e.g. C = 10,000): copy $\theta \rightarrow \theta^{-}$. Otherwise: don't touch it.



$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^{-}) - Q(s, a; \theta)\right)^2\right]$$

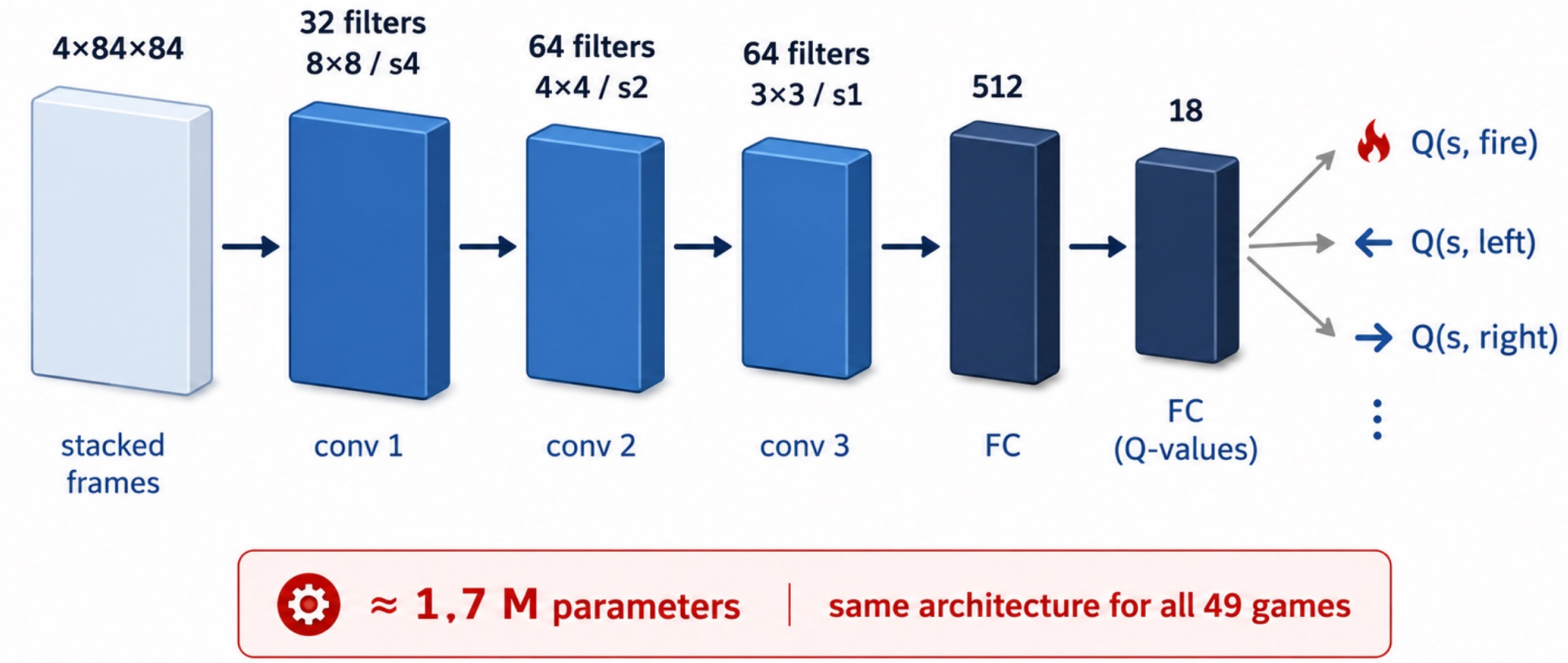
Putting It Together: The DQN Loop



Same observe–act–update skeleton from L31. Two new pieces: the buffer and the frozen copy.

Anatomy of the DQN Network

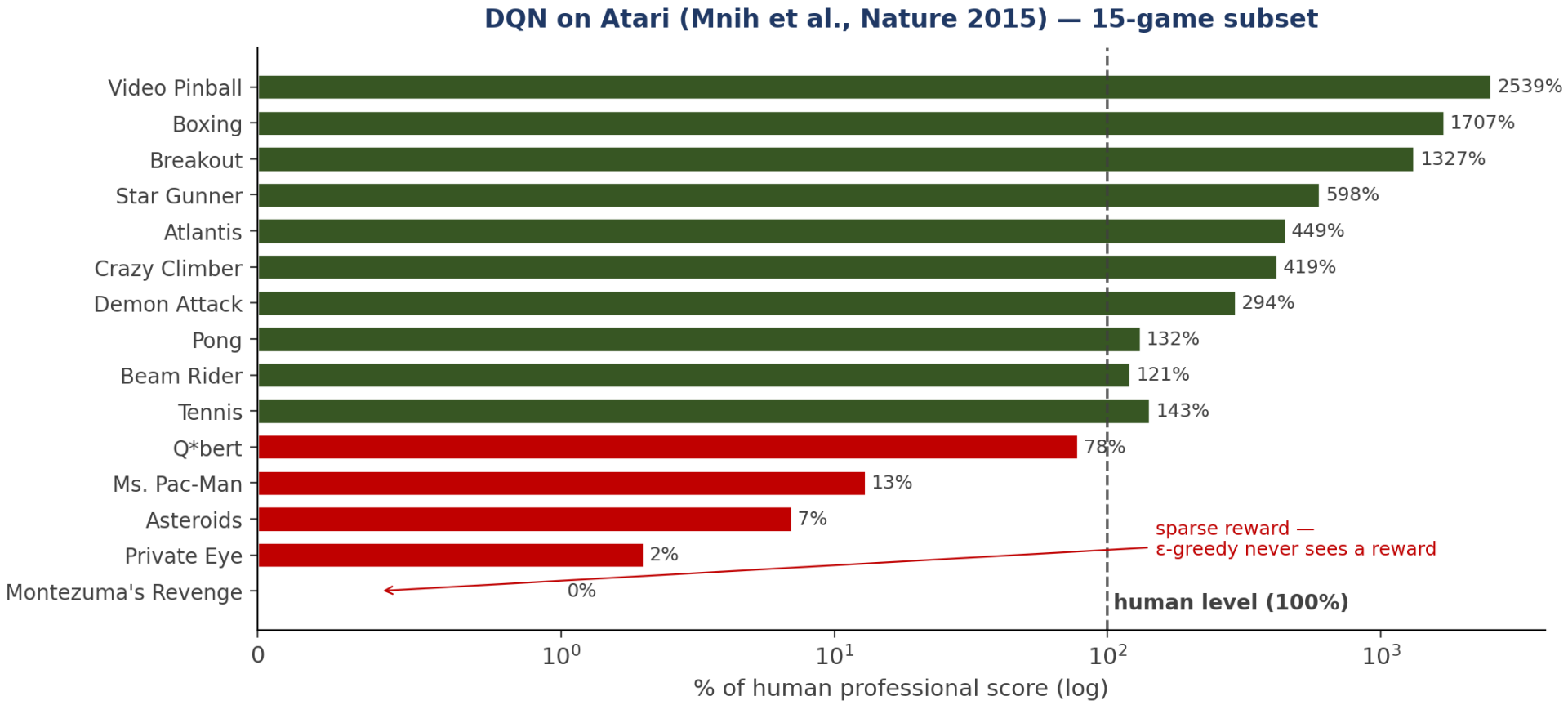
- *Mnih et al., Human-level control through deep reinforcement learning, Nature 518, 2015.*



3. The Atari Moment

Nature 2015 — The Result

- A single algorithm + a single architecture matched or exceeded a professional human tester on 29 of 49 Atari games — from pixels.



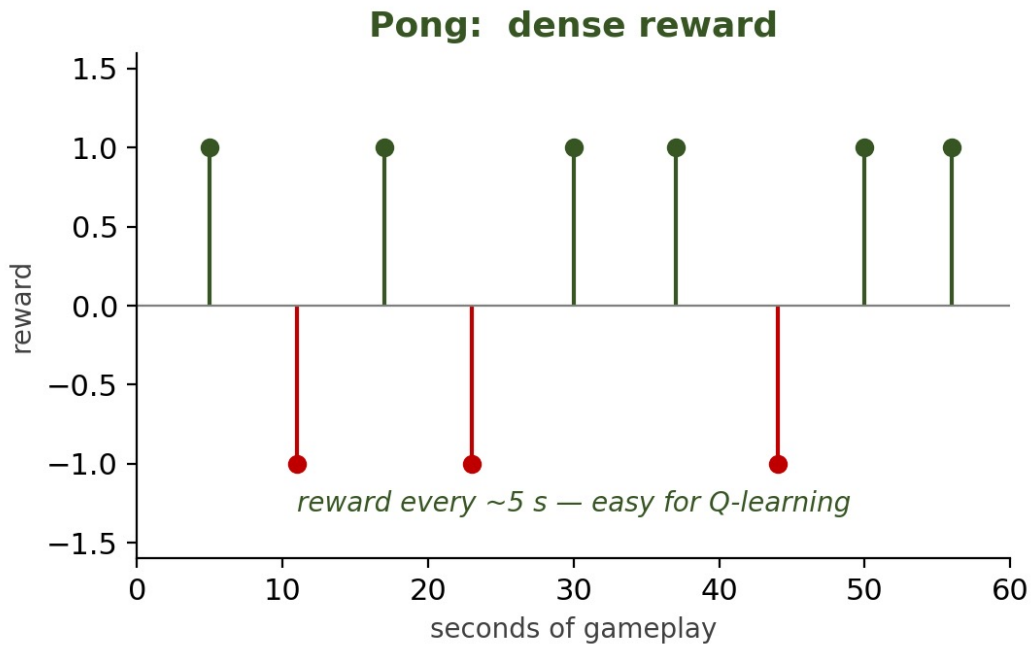
Why This Was a Big Deal

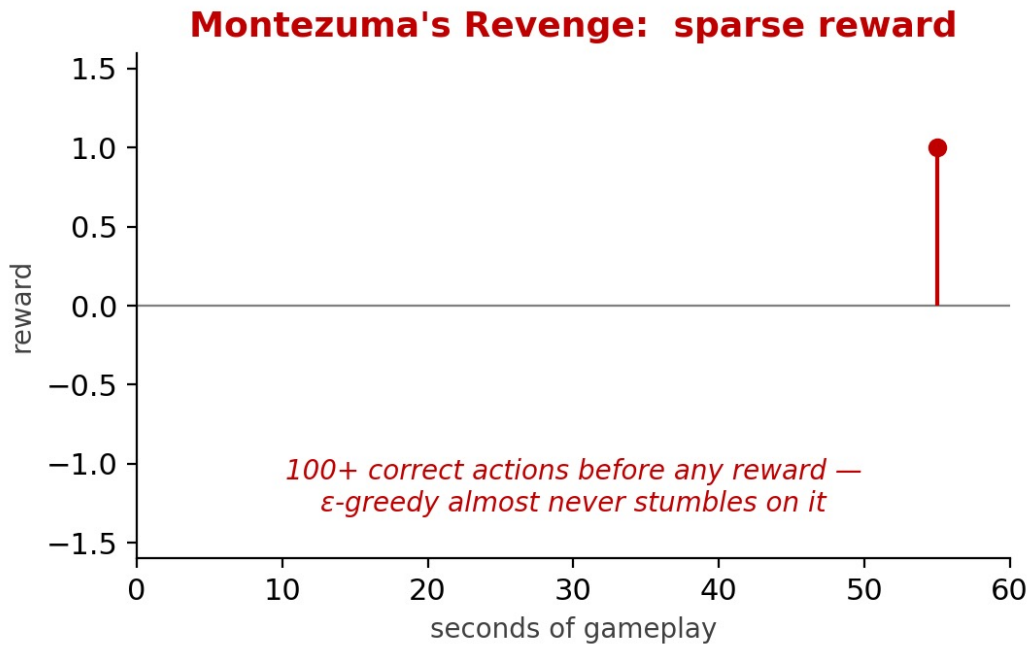
- **Pixels in.** No hand-engineered features. The CNN learns them.
- **One algorithm.** No per-game tuning. Same hyperparameters across all 49 games.
- **Game-changer for the field.** RL went from "interesting in theory" to "front page of *Nature*" in one paper.

Before this, "RL from raw perception" was a research dream. After this, it was a benchmark.

Think: Where Will DQN Fail?

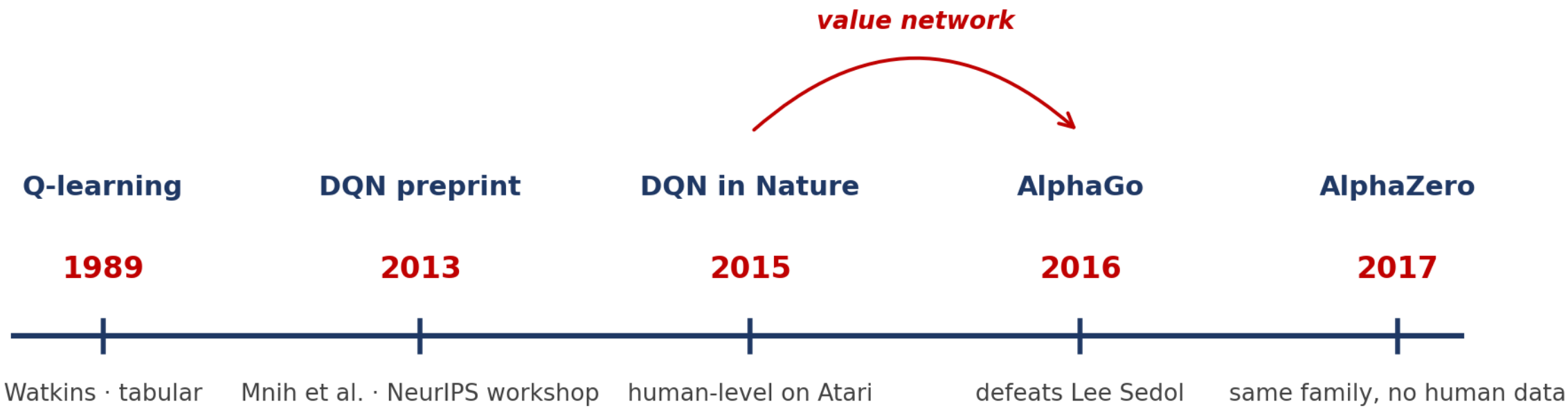
DQN is great at Pong, terrible at Montezuma's Revenge. Why?





What property of the reward signal does Q-learning depend on? When does ε-greedy stop being enough?

DQN's Lineage



From 1989 tabular Q-learning to 2017 AlphaZero — one family.

AlphaGo: DQN's Descendant

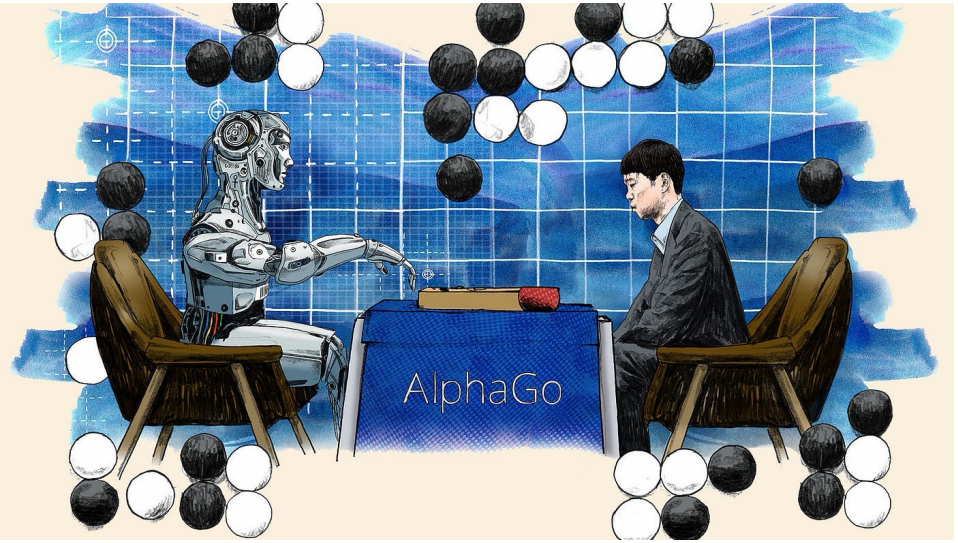
AlphaGo replaced the Q-table with two networks — a policy net and a **value net.**

- *The value net does what DQN's Q-net does — estimates how good a state is.*

- **Same bootstrapping idea:**
  train V(s) toward outcomes predicted by deeper search.

- **Same scaling reason:**
  Go has ~10^{170} states. No table will ever hold them.

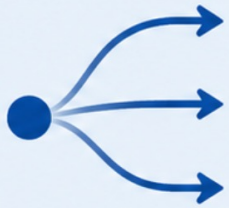
- **One key difference:**
  AlphaGo has a known world model (rules of Go), so it can also do tree search. DQN can't.



Summary

- Tabular Q-learning can't scale; Atari has more states than there are atoms in the universe.
- Replace the table with a neural net: $Q(s, a; \theta)$. Q-learning becomes regression.
- Naïve neural Q-learning is unstable for two reasons: correlated samples, moving targets.
- Experience replay fixes correlation. Target network fixes the moving target.
- DQN's value-net idea is what's inside AlphaGo and most modern game-playing systems.

Frontier: Where DQN Stops



Continuous actions

max Q(s, a) over a continuous action space is intractable.

→ *actor-critic methods*

Sparse / deceptive rewards

ε-greedy never finds the goal.

→ *intrinsic motivation, curiosity*

Sample efficiency

DQN needed 200M frames to learn Breakout. A child learns in minutes.

→ *model-based RL (L34)*

Bridge to L33: Policy Gradient

DQN learns a value, then acts greedily.

What if we learn the policy directly?

Two questions for next lecture:

- How do you "do gradient descent" on a probability distribution over actions?
- What do you do when the action space is continuous and *max* makes no sense?

L33: Policy Gradient Methods — a different family entirely.