From DQN to Double DQN

A technical comparison of representation learning and target correction

PAPER 01

DQN

The representation breakthrough. Makes value learning from raw frames practical by mapping pixels to values.

PAPER 02

Double DQN

Target correction. Decouples action selection and evaluation to repair over optimistic bias.

DOMAIN

ATARI 2600

The benchmark domain for general reinforcement learning. Control directly from pixels.

Why Atari, and Why These Papers?

BENCHMARK: ATARI 2600

High-Dim. Complexity

- Raw pixels (210×160) as state input
- Implicit motion requires frame stacking
- Sparse, delayed rewards complicate credit assignment

PAPER 01: DQN

Representation

Makes end-to-end deep learning from pixels feasible.

Representation Learning

PAPER 02: DOUBLE DQN

Target Bias

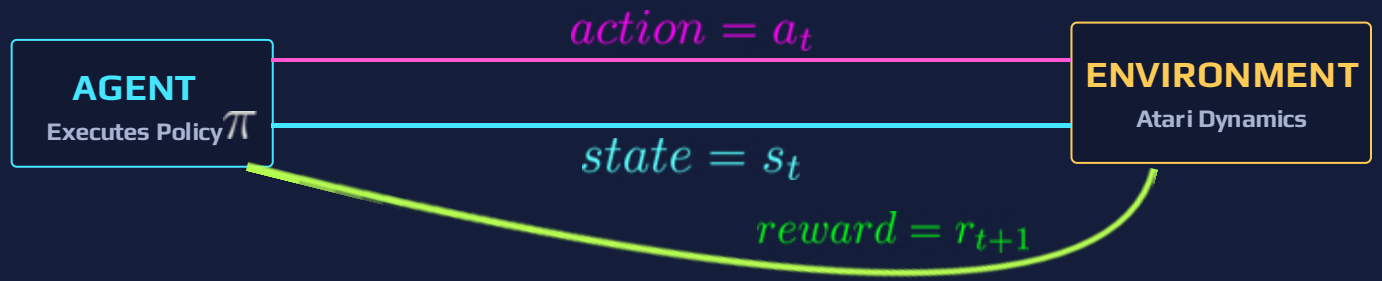
Repairs overoptimistic targets within the DQN framework.

Target Bias Correction

DQN enables **practical deep Q-learning** through robust representation, while Double DQN ensures **training stability** by improving target trustworthiness.

The Reinforcement Learning Baseline

Both papers inherit the same objective: to estimate long-term action value under discounted return.



Objective: Maximize the discounted return $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$

The Loop: Agent observes state s_t , selects action a_t following policy π , receives reward r_{t+1} and next state s_{t+1}.

The Q-Learning Update Rule

Tabular Q-learning approximates the optimal action-value function by moving current estimates toward a one-step Bellman target.

Current Estimate
The value we are trying to refine.

Learning Rate (α)

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right]$$

One-Step Bellman Target: The immediate reward plus the discounted maximum value of the next state.

Bootstrapping: Learning from Estimates

The update rule relies on **bootstrapping**—updating an estimate based on another estimate. While this allows learning without full episodes, it can propagate approximation errors through the network.

Why Atari is a Hard Testbed for RL

Atari combines high-dimensional visual perception with delayed-reward control problems.

High-Dimensional Input

210×160 RGB pixels as state input instead of compact symbolic vectors.

Partial Observation

A single frame cannot reveal motion direction or velocity; requires frame stacking.

Sparse, Delayed Rewards

Long gaps between actions and consequences complicate credit assignment.

Correlated Samples

Consecutive video frames are nearly identical, breaking the i.i.d. assumption.

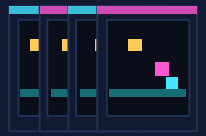
Non-Stationary Data

As the agent learns, the distribution of visited states and actions shifts constantly.

DQN: Architecture and State Representation

The first paper keeps the Q-learning objective but replaces the tabular value function with a convolutional approximator.

PIPELINE

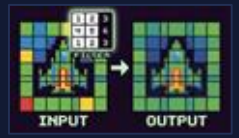


last 4 frames

84×84×4 stack

Conv 1

16 filters
8×8, stride 4



Conv 2

32 filters
4×4, stride 2

FC

256 ReLU units

Q-values for all actions

Single forward pass predicts every action value at once.

Frame Stacking Rationale

A single frame often does not reveal motion direction or velocity. Stacking provides the temporal context needed for state representation.

Training Objective

DQN training objective optimizes for Bellman error via SGD.

DQN Works Because of the Training Recipe Around the Network

Replay memory and off-policy updates are as important as the convolutional encoder itself.

Training loop

1. Observe frame stack and preprocess 4 frames
2. Choose an action with an ε-greedy policy
3. Store the transition in replay memory
4. Sample a random minibatch from replay
5. Update the network with SGD on the TD loss

Why replay works

- Breaks the short-range correlation in consecutive frames
- Reuses older transitions across many updates
- Smooths the training distribution over many earlier policies

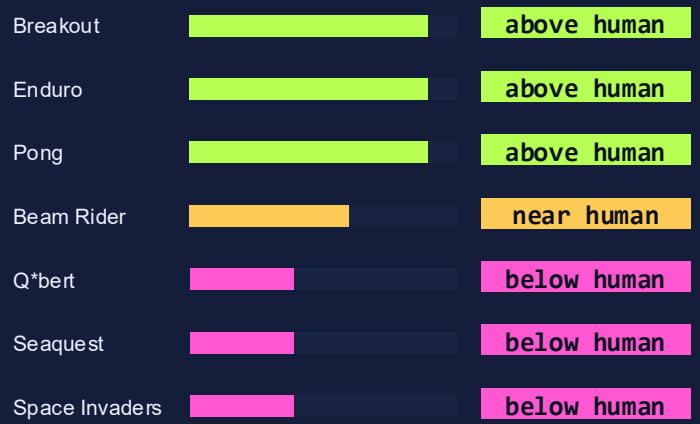
Important compromises

- Rewards are clipped to $\{-1, 0, 1\}$
- Uniform replay does not prioritize informative transitions
- The setup is optimized for trainability, not perfect value calibration

What DQN Proves and What It Does Not Prove

The first paper is foundational because it proves feasibility, but its success does not guarantee clean targets.

SEVEN-GAME SUMMARY



Reconstructed from the DQN paper's seven-game evaluation: 3 above human, 1 near human, and 3 clearly below human.

MY READ

Why the paper matters

- It proves that one shared architecture can learn directly from raw Atari pixels
- It joins representation learning and control instead of separating them
- It shows that replay-based deep value learning can be made practical at scale

What it does not settle

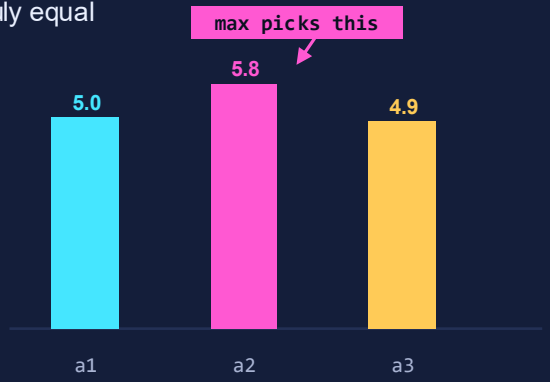
- Some longer-horizon games still remain far from human performance
- Good behavior does not prove the Q-values themselves are unbiased or well calibrated
- The paper does not isolate whether remaining instability comes from representation error or target error

The Problem: DQN Can Overestimate Action Values

The max operator favors actions whose estimates are spuriously high, and bootstrapping can propagate that optimism.

INTUITION

Toy example: all three actions are truly equal



Even if the errors average out overall, the maximization step tends to keep the lucky overestimate.

PAPER ARGUMENT

$$\max_a Q_t(s,a) \geq V^*(s) + \sqrt{\frac{C}{m-1}}$$

- The paper shows that even unbiased action-level errors can become a positively biased max estimate
- This is not a helpful exploration bonus; it is accidental overoptimism caused by the target
- Because Q-learning bootstraps, an optimistic target can contaminate later targets and eventually policy quality

Translation into plain English: DQN may learn a representation that is good enough to act well, but still use value estimates that are systematically too optimistic.

Double DQN Changes the Target, Not the Whole Framework

The idea is minimal: let one network choose the action, and let the target network evaluate it.

DQN TARGET

$$Y_t^{DQN} = R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_t^-)$$

One estimator is doing both jobs

- The maximizing action is selected from the same value estimates used in the target
- That coupling is what makes positive errors especially likely to survive the max step

DOUBLE DQN TARGET

$$Y_t^{DoubleDQN} = R_{t+1} + \gamma Q\left(S_{t+1}, \arg\max_a Q(S_{t+1}, a; \theta_t); \theta_t^-\right)$$

online network chooses action → argmax → **target network** evaluates action

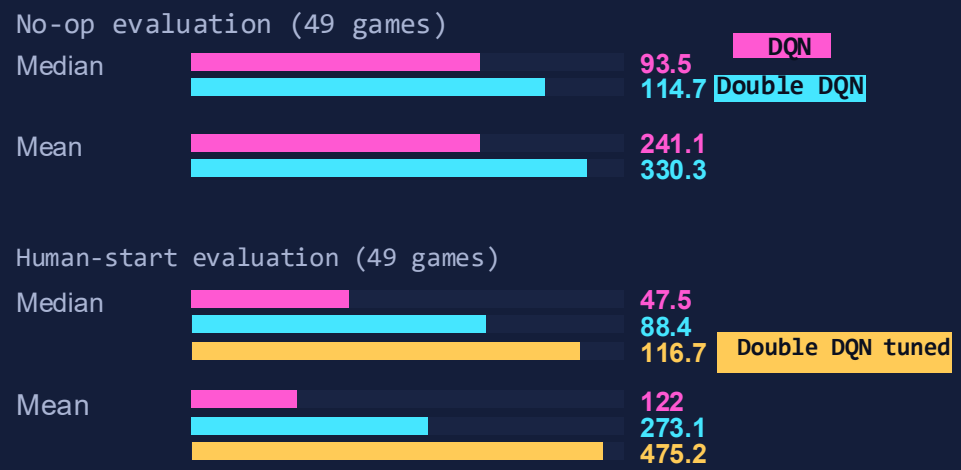
Minimal change, cleaner statistical job.

The key point is that Double DQN keeps replay memory, stacked frames, off-policy learning, and the CNN family fixed. The meaningful edit is the target definition itself.

The Empirical Case: More Accurate Values and Stronger Policies

The second paper argues that reduced overestimation improves both calibration and control quality.

NORMALIZED PERFORMANCE



Interpretation: the gains are not just cosmetic. They show up in policy-level evaluation.

KEY CLAIMS

- Value estimates in DQN can overshoot the true discounted return
- In some games, DQN's value spikes coincide with actual score collapse
- Double DQN is more stable and often produces better policies

Highlighted game-level improvements

- Road Runner: 233% → 617%
- Asterix: 70% → 180%
- Zaxxon: 54% → 111%
- Double Dunk: 17% → 397%

The Clean Comparison: What Changed, What Did Not, and Why It Matters

The pairing works because the second paper keeps the framework fixed while correcting the estimator.

WHAT STAYED THE SAME

- Value-based, off-policy deep reinforcement learning
- Atari control from stacked raw frames
- Bootstrapping, replay memory, and convolutional function approximation

WHAT CHANGED

- The target definition
- DQN uses one estimator too aggressively inside the max step
- Double DQN lets the online network select and the target network evaluate

WHY IT MATTERS

- DQN is the representation breakthrough
- Double DQN is the target-calibration correction
- The method story moves from feasibility to reliability

FEASIBILITY → CALIBRATION

Conclusion

The larger lesson is that deep reinforcement learning matured through both stronger representations and better targets.

1 DQN proves the idea

- Raw pixels can support value-based control
- CNN features and control can be learned jointly
- Experience replay makes the optimization tractable

2 Double DQN repairs the target

- Overestimation comes from the max step over noisy estimates
- Selection and evaluation are decoupled
- Learning becomes more stable and often more accurate

3 Together, the papers tell one method story

- Feasibility comes first
- Calibration comes next
- Progress is about both representation and estimator design

Final takeaway: DQN makes deep value learning from Atari pixels practical; Double DQN makes that same value learning more trustworthy.