Human-Level Control through Deep Reinforcement Learning

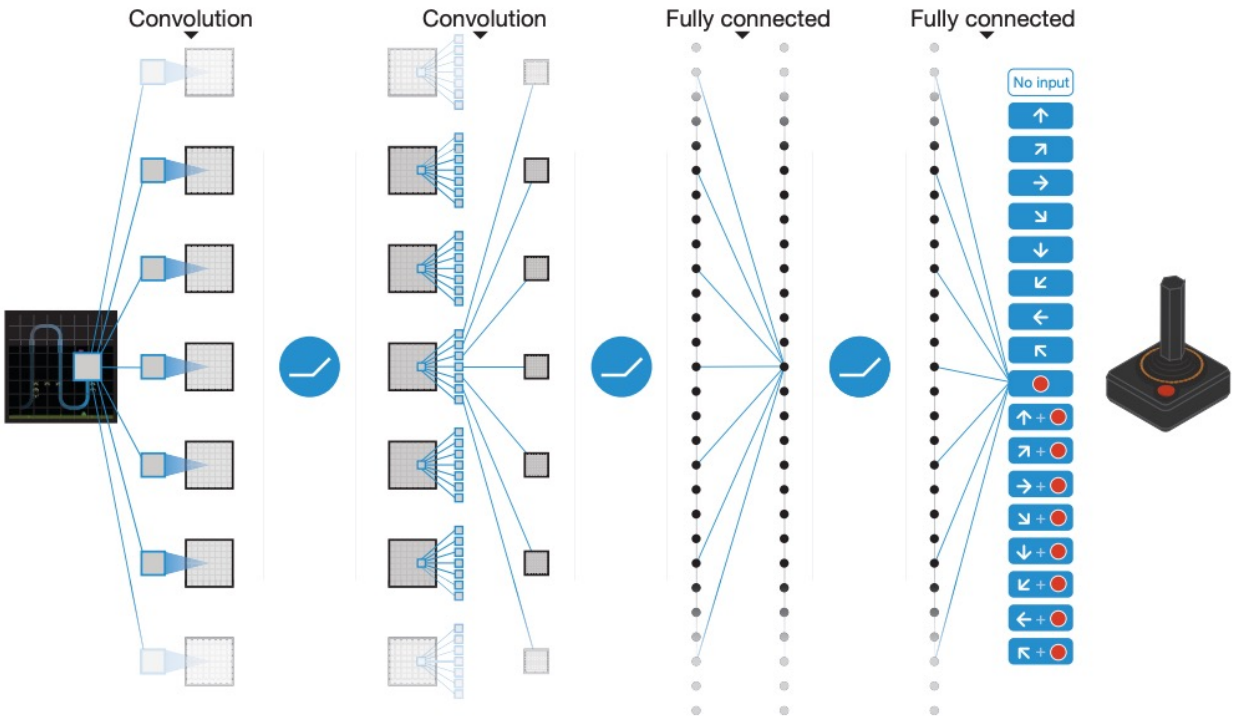
V. Mnih, K. Kavukcuoglu, D. Silver, et al.

Nature (2015)

Background & Motivation

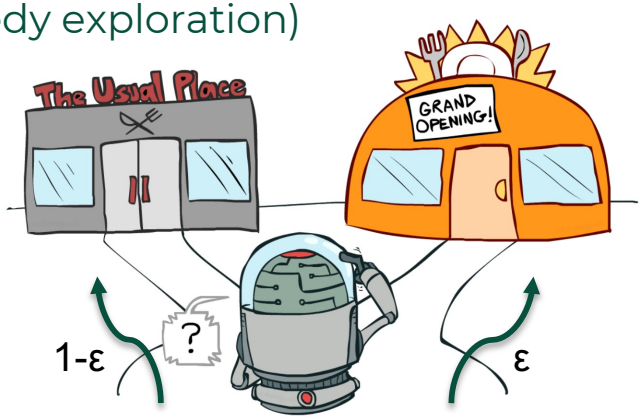
- Reinforcement Learning (RL): learn by trial-and-error to maximize reward
- Low-dimensional inputs are handled with static Q-tables
- Challenge: high-dimensional inputs (e.g. raw images) + unstable training with deep nets
- Prior approaches: handcrafted features + linear models
- DQN: learn directly from pixels on nonlinear models

DQN Architecture on Atari games (Input → Features → Q-values)



Q-Learning

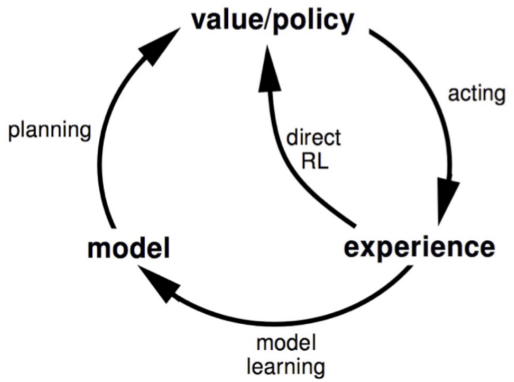
- Q(s,a): expected discounted return from state s taking action a
- Bellman optimality: $Q^*(s,a) = E_{s'}\quad Q^*(s,a) \quad = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q^*(s',a') | s,a\right]$
- Iterative update: $Q_{i+1}(s,a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q_i(s',a') | s,a]$.
  - *For this case you have to go through all possible state action values, inefficient -> function approximator for Q*
- DQN learns $Q(s,a;\theta) \approx Q^*(s,a)$ with a deep network
- Policy during control $a_t = argmax_a Q(s_t, a; \theta)$(with ε-greedy exploration)



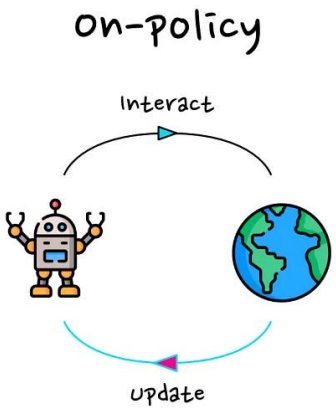
Theoretical components

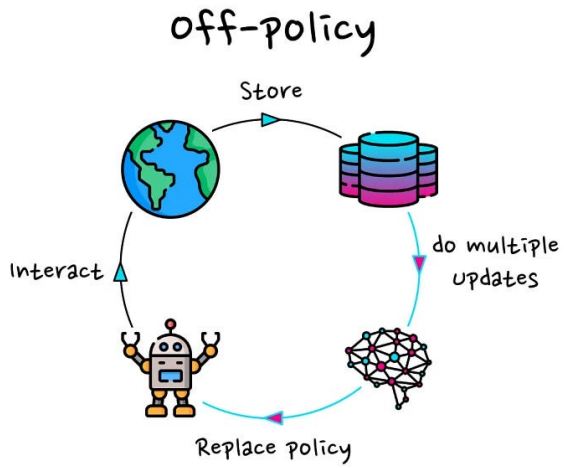
DQN is

- Model-free: learns V/Q/π directly from samples



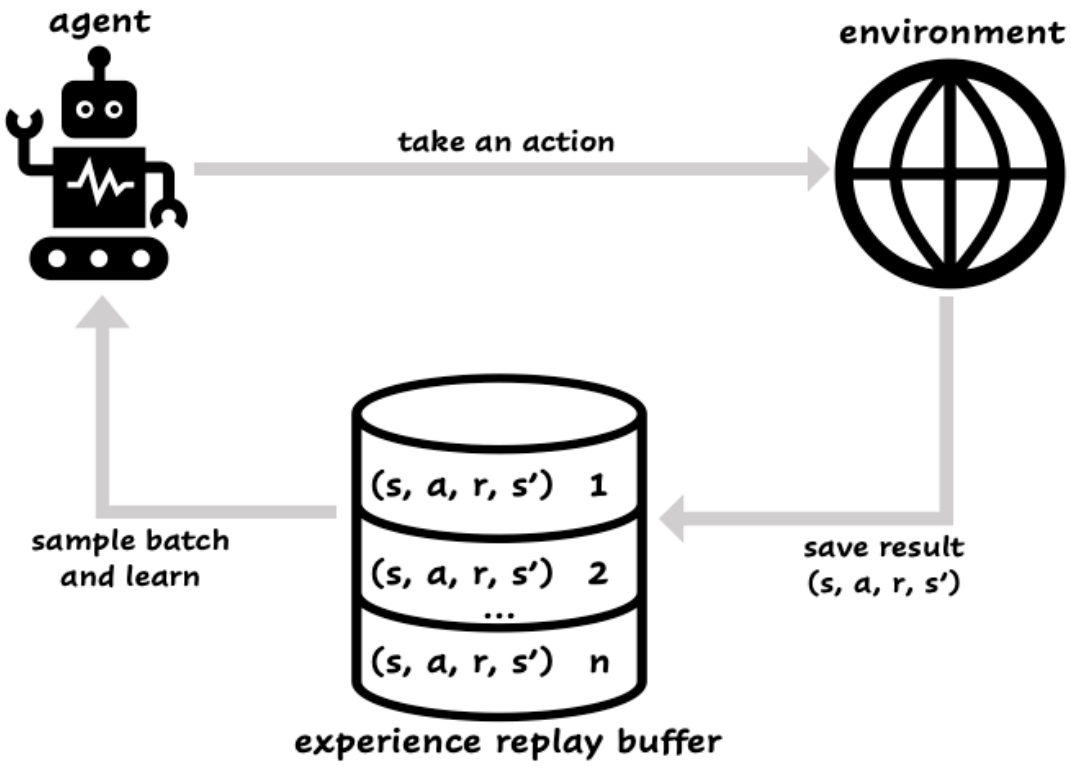
- Off-policy based: it learns a value function for a target (greedy) policy while data come from a different behavior policy (e.g., ε-greedy with replay).





Stabilization

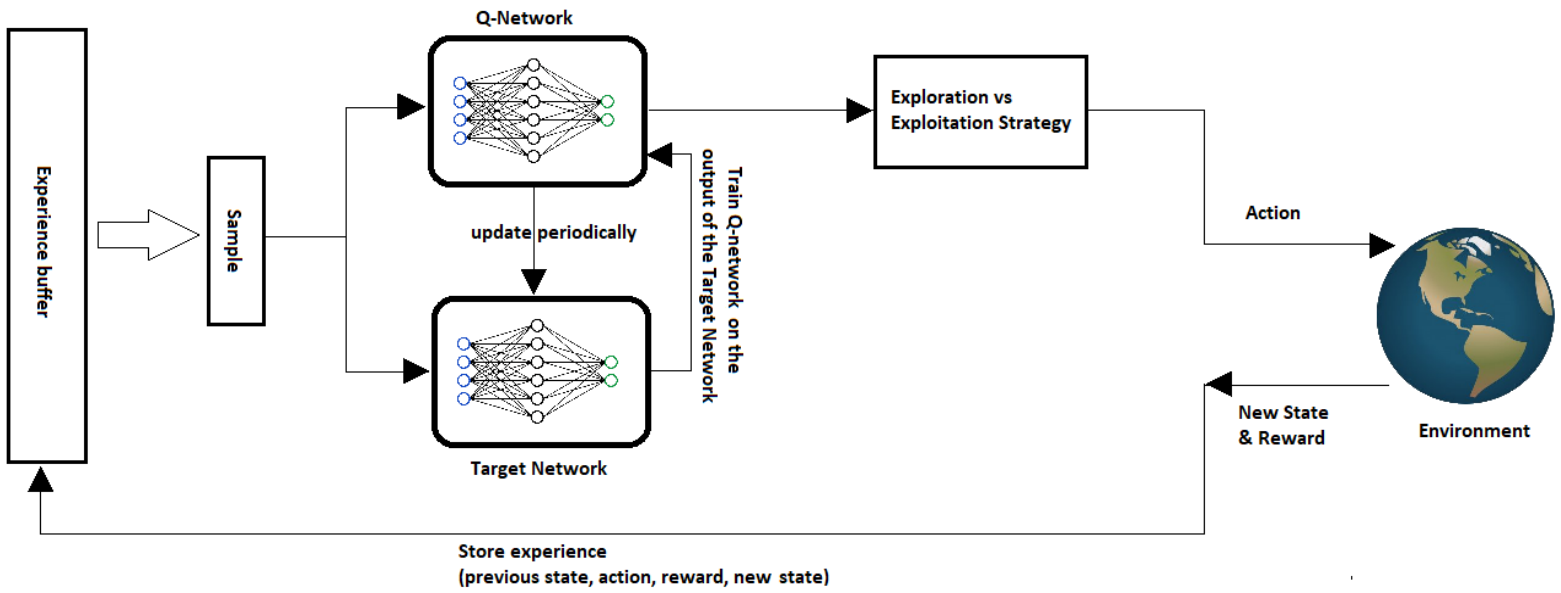
- Experience Replay: store (s,a,r,s') and sample random minibatches during updating parameters:
  - Breaks temporal correlation
  - Improves sample efficiency via reuse



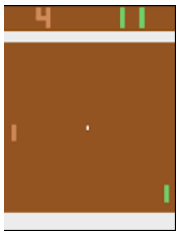
Stabilization

- Target Network θ^-: compute targets with a frozen copy of θ
  - $y = r + \gamma \, max_{a'} Q(s', a';\, \theta^-)$
  - $\mathrm{L} = y \; - Q(s, a; \theta_i)$, sync $\theta^- \leftarrow \theta$ every C

Updates:



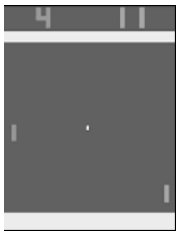
Preprocessing



Input
210×160×RGB
with 128-color
palette
Take per pixel
max for Frame I
and i-1 to
remove sprite
flicker

Convert RGB to
luminance (Y).
This drops colour

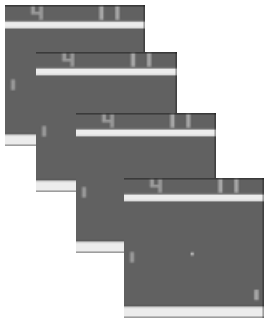
but keeps
intensity/edges.
210x160x1images



Down sample

the grayscale
image to 84×84

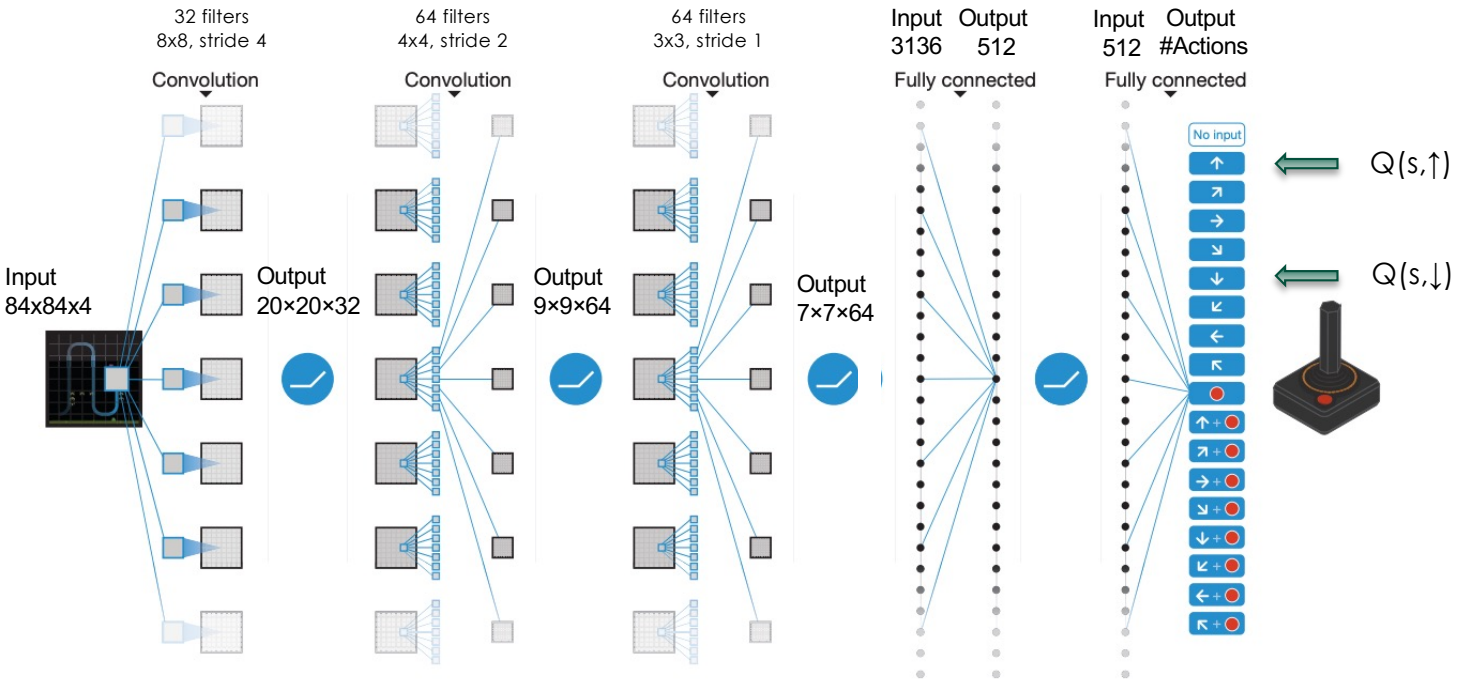
Take last 4
frames



Input of DQN model

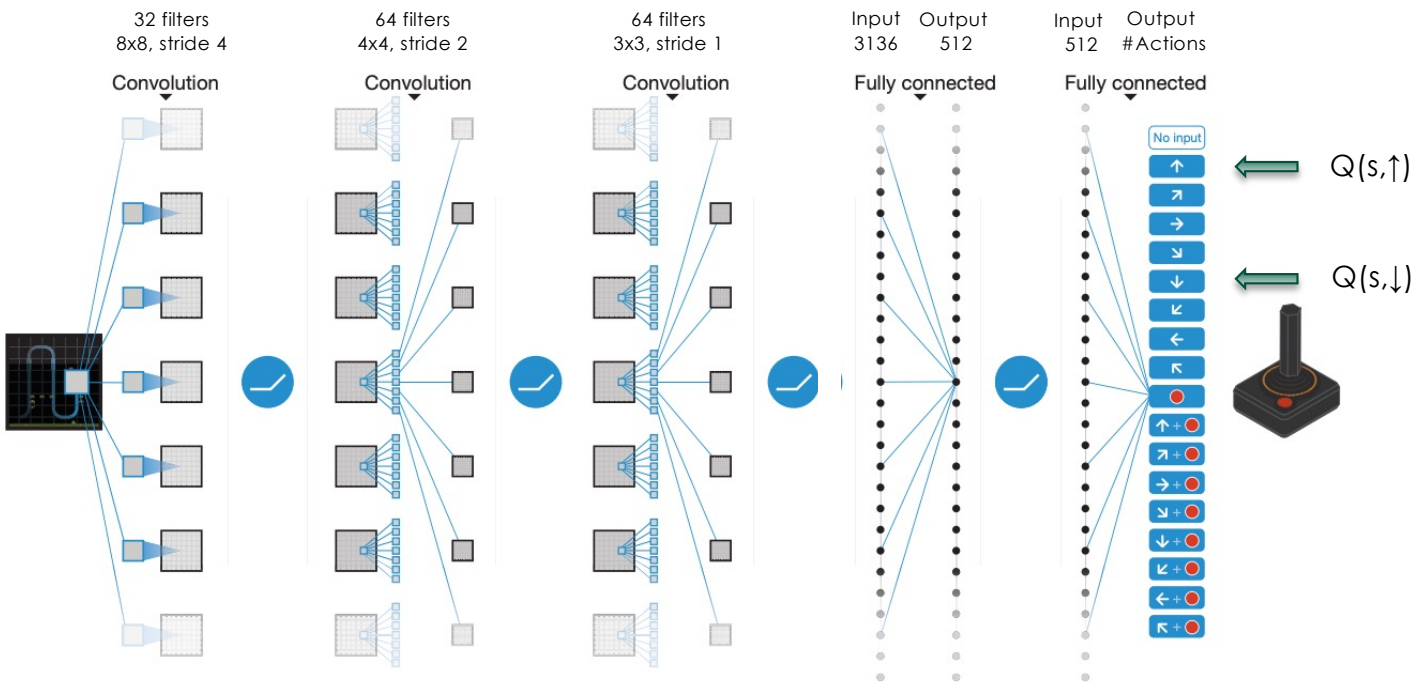
DQN Architecture on Atari games

(Input → Features → Q-values)



DQN Architecture on Atari games

(Input → Features → Q-values)



Algorithm

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$

For episode = 1, M **do**

  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

  For $t = 1$,T **do**

    With probability ε select a random action a_t

    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$

    Execute action a_t in emulator and observe reward r_t and image x_{t+1}

    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

    Store transition $\left(\phi_t, a_t, r_t, \phi_{t+1}\right)$ in D

    Sample random minibatch of transitions $\left(\phi_j, a_j, r_j, \phi_{j+1}\right)$ from D

    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^{-}\right) & \text{otherwise} \end{cases}$

    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ

    Every C steps reset $\hat{Q} = Q$

  End For

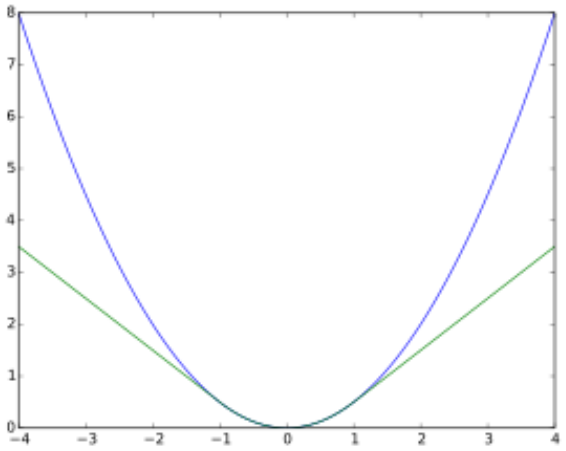
End For

Reward & TD-error clipping

- Reward clipping: map rewards to {−1,0,1}→ normalized scale across games.
  One LR works everywhere.

- TD error: $r+\gamma \max_{a'} Q\left(s',a';\theta_i^-\right)-Q(s,a;\theta_i)$

  Clip error δ to [−1,1]: acts like Huber loss (L2 near 0, L1 for |δ|>1) → caps per-sample gradient.

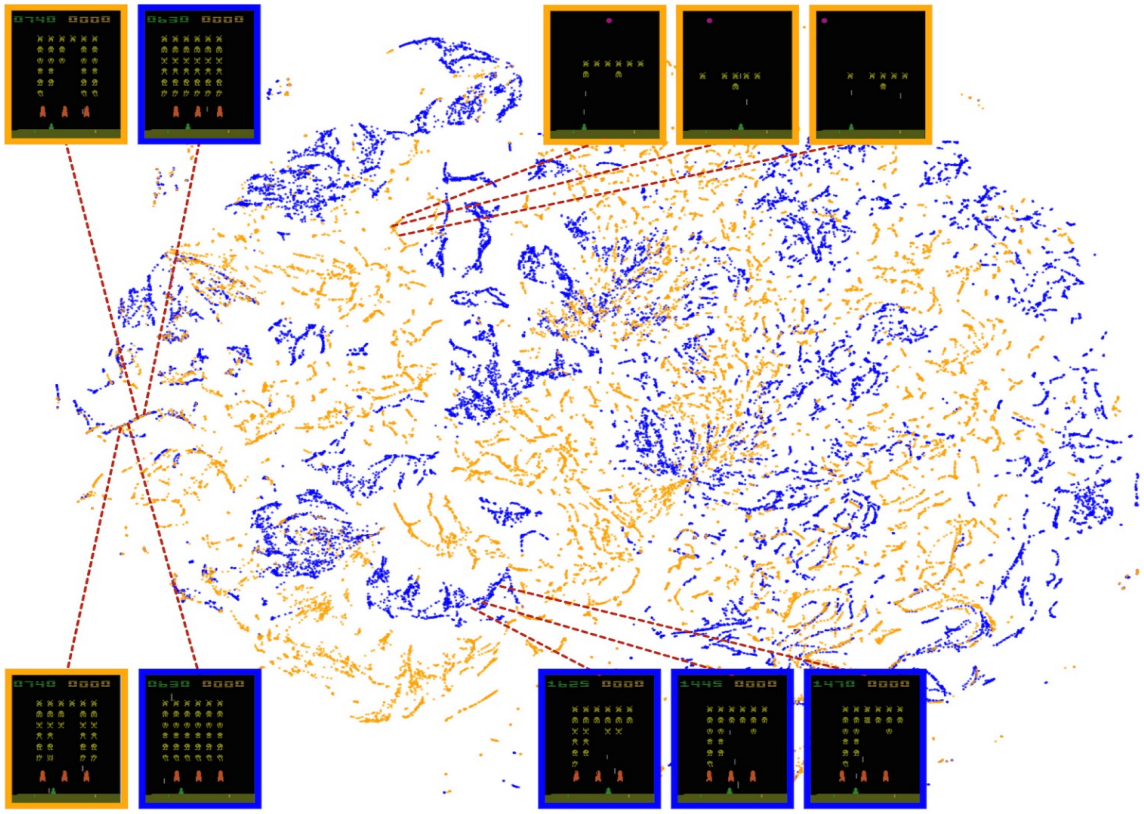


Training Protocol (Atari)

Hyperparameter	Value	Description
minibatch size	32	Number of training cases over which each stochastic gradient descent (SGD) update is computed.
replay memory size	1000000	SGD updates are sampled from this number of most recent frames.
agent history length	4	The number of most recent frames experienced by the agent that are given as input to the Q network.
target network update frequency	10000	The frequency (measured in the number of parameter updates) with which the target network is updated (this corresponds to the parameter *C* from Algorithm 1).
discount factor	0.99	Discount factor gamma used in the Q-learning update.
action repeat	4	Repeat each action selected by the agent this many times. Using a value of 4 results in the agent seeing only every 4th input frame.
update frequency	4	The number of actions selected by the agent between successive SGD updates. Using a value of 4 results in the agent selecting 4 actions between each pair of successive updates.
learning rate	0.00025	The learning rate used by RMSProp.
gradient momentum	0.95	Gradient momentum used by RMSProp.
squared gradient momentum	0.95	Squared gradient (denominator) momentum used by RMSProp.
min squared gradient	0.01	Constant added to the squared gradient in the denominator of the RMSProp update.
initial exploration	1	Initial value of ε in ε-greedy exploration.
final exploration	0.1	Final value of ε in ε-greedy exploration.
final exploration frame	1000000	The number of frames over which the initial value of ε is linearly annealed to its final value.
replay start size	50000	A uniform random policy is run for this number of frames before learning starts and the resulting experience is used to populate the replay memory.
no-op max	30	Maximum number of “do nothing” actions to be performed by the agent at the start of an episode.

Visualizing DQN Representations with t-SNE

- Two-dimensional t-SNE embedding of the representations in the last hidden layer assigned by DQN to game states



Results — DQN Performance

- Linear/value-function baselines previously favored for stability but can't handle complex nonlinear relationships (e.g. SARSA)

Game	Random Play	Best Linear Learner	Contingency (SARSA)	Human	DQN (± std)	Normalized DQN (% Human)
Alien	227.8	939.2	103.2	6875	3069 (±1093)	42.7%
Amidar	5.8	103.4	183.6	1676	739.5 (±3024)	43.9%
Assault	222.4	628	537	1496	3359(±775)	246.2%
Asterix	210	987.3	1332	8503	6012 (±1744)	70.0%
Asteroids	719.1	907.3	89	13157	1629 (±542)	7.3%
Atlantis	12850	62687	852.9	29028	85641(±17600)	449.9%
Bank Heist	14.2	190.8	67.4	734.4	429.7 (±650)	57.7%
Battle Zone	2360	15820	16.2	37800	26300 (±7725)	67.6%
Beam Rider	363.9	929.4	1743	5775	6846 (±1619)	119.8%
Bowling	23.1	43.9	36.4	154.8	42.4 (±88)	14.7%
Boxing	0.1	44	9.8	4.3	71.8 (±8.4)	1707.9%
Breakout	1.7	5.2	6.1	31.8	401.2 (±26.9)	1327.2%
Centipede	2091	8803	4647	11963	8309(±5237)	63.0%
Chopper Command	811	1582	16.9	9882	6687 (±2916)	64.8%
Crazy Climber	10781	23411	149.8	35411	114103 (±22797)	419.5%
Demon Attack	152.1	520.5	0	3401	9711 (±2406)	294.2%
Double Dunk	-18.6	-13.1	-16	-15.5	-18.1 (±2.6)	17.1%
Enduro	0	129.1	159.4	309.6	301.8 (±24.6)	97.5%
Fishing Derby	-91.7	-89.5	-85.1	5.5	-0.8 (±19.0)	93.5%
Freeway	0	19.1	19.7	29.6	30.3 (±0.7)	102.4%
Frostbite	65.2	216.9	180.9	4335	328.3 (±250.5)	6.2%
Gopher	257.6	1288	2368	2321	8520 (±3279)	400.4%

Stability features importance

Game	With replay, with target Q	With replay, without target Q	Without replay, with target Q	Without replay, without target Q
Breakout	316.8	240.7	10.2	3.2
Enduro	1006.3	831.4	141.9	29.1
River Raid	7446.6	4102.8	2867.7	1453.0
Seaquest	2894.4	822.6	1003.0	275.8
Space Invaders	1088.9	826.3	373.2	302.0

Impact on AI

1

Advanced deep RL: Double DQN, Prioritized Replay, Critic-Actor Networks

2

Influenced large-scale systems (e.g., AlphaGo's use of deep nets + RL)

3

Paved the way for general game-playing and real-world control applications

Conclusions

- DQN = Q-learning + deep CNN (pixels → Q-values)
- Stabilizers (replay, target net, clipped TD loss) made deep RL practical
- Results: human-level control in many Atari games with a single agent architecture
- Impact: breakthrough in DRL with many successors and applications
- Takeaway: smart training design overcame instability.