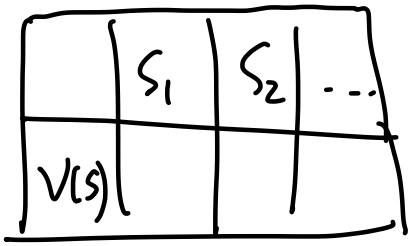
6. Value function approximation

6.1 large problem

① go 10^{170} states

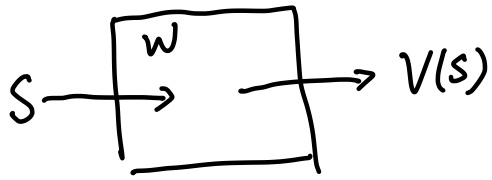
② Helicopter continuous states

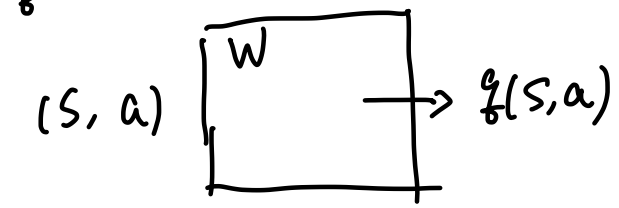


Impossible list all states in a table

How do scale up?

$$V(S) \Longleftrightarrow V(S, w)$$





update w instead of V, q

6.2 Approximation function

① Linear function

② NN

6.3 Incremental Method

6.3.1 Gradient Descent (GD)

Define loss

$$J(w) = E_\pi\left[\left(V_\pi(S) - \hat{V}(S,w)\right)^2\right]$$

Supervised learning $V_\pi(s)$: real $\quad \hat{V}(s,w)$: approximation

$$w^* = \arg\min_w J(w)$$

GD

$$\Delta w = -\frac{\alpha}{2}\nabla_w J(w) = \alpha E_\pi\left[\left(V_\pi(s) - \hat{V}(s,w)\right)\nabla_w \hat{V}(s,w)\right]$$

SGD Batch 1

$$\Delta w = \alpha\left(V_\pi(s_i) - \hat{V}(s_i,w)\right)\nabla_w \hat{V}(s_i,w)$$

Batch $|B|$

$$\Delta w = \frac{1}{|B|}\sum \Delta w_i$$

6.3.2 Linear combination of feature

$$\hat{V}(s,w) = x(s)^T w = \sum_{j=1}^{n} x_j(s) w_j$$

$$\Delta w = \alpha\left(V_\pi(s) - \hat{V}(s,w)\right)x(s)$$

If $x(s) = \begin{pmatrix} 1(s=s_1) \\ 1(s=s_2) \\ \vdots \\ 1(s=s_n) \end{pmatrix}$ previous Table Method

6.3.3 Study $V_\pi(s)$

① MC $\Delta w = \alpha(G_t - \hat{v}(S_t,w)) \nabla_w \hat{v}(S_t,w)$

② TD(0) $\Delta w = \alpha(R_{t+1} + \gamma \hat{v}(S_{t+1},w) - \hat{v}(S_t,w)) \nabla_w \hat{v}(S_t,w)$

③ TD(λ) $\Delta w = \alpha(G_t^\lambda - \hat{v}(S_t,w)) \nabla_w \hat{v}(S_t,w)$

$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$

$V(s) = E(G_t | S_t = s)$

$G_t^{(1)} = R_{t+1} + \gamma V(S_{t+1})$

$\vdots$

$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$

$\vdots$

$G_t^\lambda = (1-\lambda) \sum_{n=1}^{\infty} G_t^{(n)} \lambda^{n-1}$

Dataset

For MC: $\{\langle S_i, G_i \rangle\}_{i=1}^{T}$

TD: $\{\langle S_i, R_{i+1} + \gamma V(S_{i+1}) \rangle\}_{i=1}^{T-1}$

TD(λ): $\{\langle S_i, G_t^\lambda \rangle\}_{i=1}^{T-1}$

Very important problem

This is not an exact GD?

Why?

$$J(w) = E_\pi\left[\left(V_\pi(S) - \hat{V}(S,w)\right)^2\right]$$

TD(0) $$J(w) = E_\pi\left(\left(R_{t+1} + \gamma \hat{V}(S_{t+1},w) - \hat{V}(S,w)\right)^2\right)$$

Exact GD, Gradient on $\hat{V}(S_{T+1},w)$, $\hat{V}(S,w)$

6.4 Control problem

$$\hat{q}(S,A,w) \longrightarrow q_\pi(S,A)$$

$$J(w) = E_\pi\left[\left(q_\pi(S,A) - \hat{q}(S,A,w)\right)^2\right]$$

For MC: $\Delta w = \alpha\left(G_t - \hat{q}(S,A,w)\right)\nabla_w \hat{q}(S,A,w)$

TD(0): $\Delta w = \alpha\left(R_{t+1} + \gamma \hat{q}(S_{t+1},A_{t+1},w) - \hat{q}(S_t,A_t,w)\right)\nabla_w \hat{q}(S_t,A_t,w)$

TD(λ): $\Delta w = \alpha\left(q_t^\lambda - \hat{q}(S_t,A_t,w)\right)\nabla_w \hat{q}(S_t,A_t,w)$

Convergence of Control Algorithms

Algorithm	Table Lookup	Linear	Non-Linear
Monte-Carlo Control	✓	(✓)	✗
Sarsa	✓	(✓)	✗
Q-learning	✓	✗	✗
Gradient Q-learning	✓	✓	✗

(✓) = chatters around near-optimal value function

6.5 Replay or not

$D = \{ \langle s_1, V_1^\pi \rangle, (s_2, V_2^\pi), \cdots, (s_T, V_T^\pi) \}$

Not Replay

$\langle s_i, V_i^\pi \rangle$ one by one following Time order

Replay every time randomly select $\langle s_i, V_i^\pi \rangle$ from D

Apply (S)GD

6.6 Many states, but relative few actions

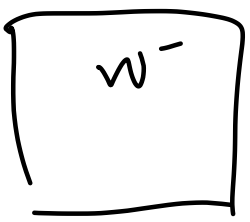
Experience Replay in Deep Q-Networks (DQN)

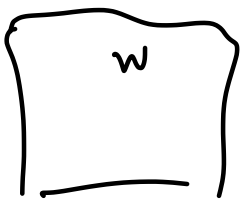
DQN uses experience replay and fixed Q-targets

- Take action a_t according to ϵ-greedy policy
- Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in replay memory $\mathcal{D}$
- Sample random mini-batch of transitions (s, a, r, s') from $\mathcal{D}$
- Compute Q-learning targets w.r.t. old, fixed parameters w^-
- Optimise MSE between Q-network and Q-learning targets

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

- Using variant of stochastic gradient descent





$\cancel{1}$ 1 : $\tilde{w} = w$

$w \leftarrow w + \Delta w$ $\quad \Delta w = \nabla_w L(\tilde{w}, w)$

⋮

K $\quad w \leftarrow w + \Delta w$ $\quad \Delta w = \nabla_w L(\tilde{w}, w)$

k+1 $\quad \tilde{w} = w$

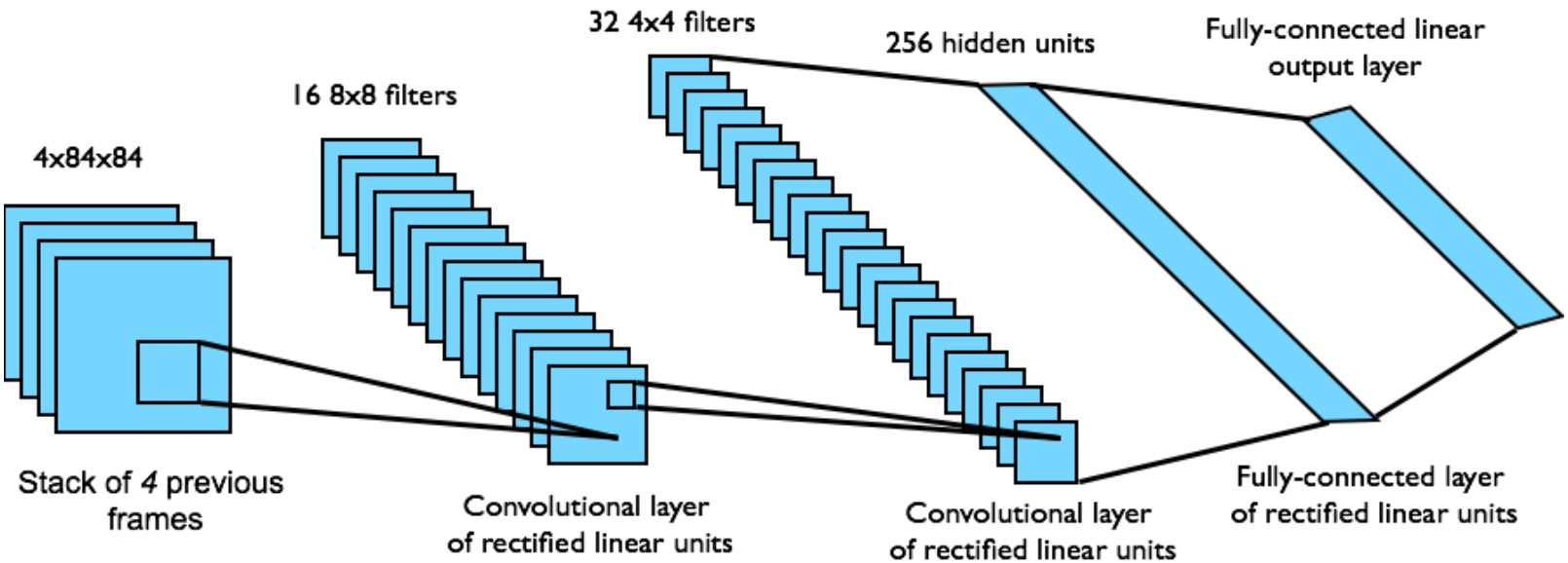
$w = w + \Delta w$ $\quad \Delta w = \nabla_w L(\tilde{w}, w)$

⋮

$w = \cdots$ $\quad \cdots$

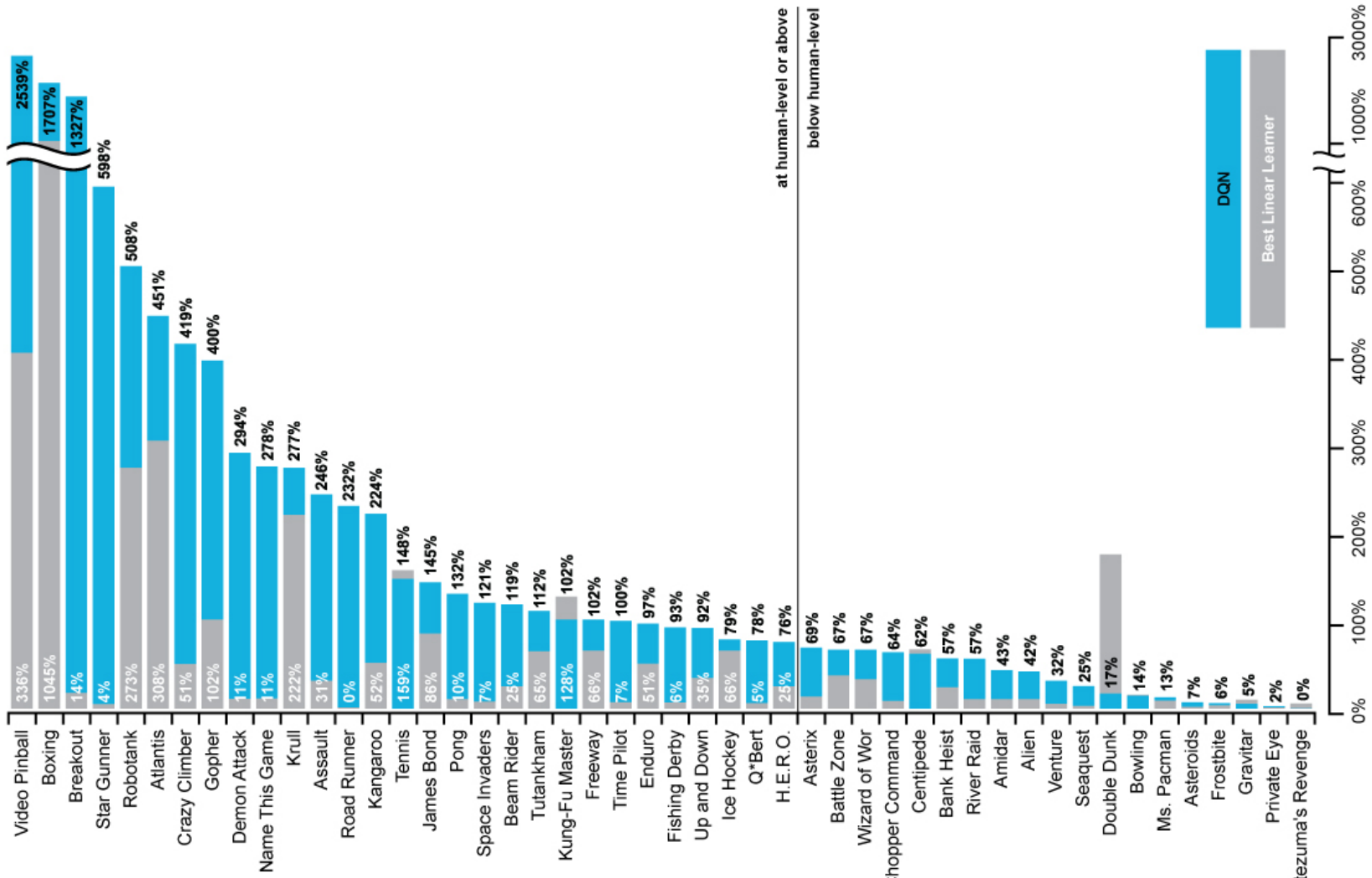
DQN in Atari

- End-to-end learning of values $Q(s, a)$ from pixels s
- Input state s is stack of raw pixels from last 4 frames
- Output is $Q(s, a)$ for 18 joystick/button positions
- Reward is change in score for that step



Network architecture and hyperparameters fixed across all games

DQN Results in Atari



How much does DQN help?

	Replay Fixed-Q	Replay Q-learning	No replay Fixed-Q	No replay Q-learning
Breakout	316.81	240.73	10.16	3.17
Enduro	1006.3	831.25	141.89	29.1
River Raid	7446.62	4102.81	2867.66	1453.02
Seaquest	2894.4	822.55	1003	275.81
Space Invaders	1088.94	826.33	373.22	301.99