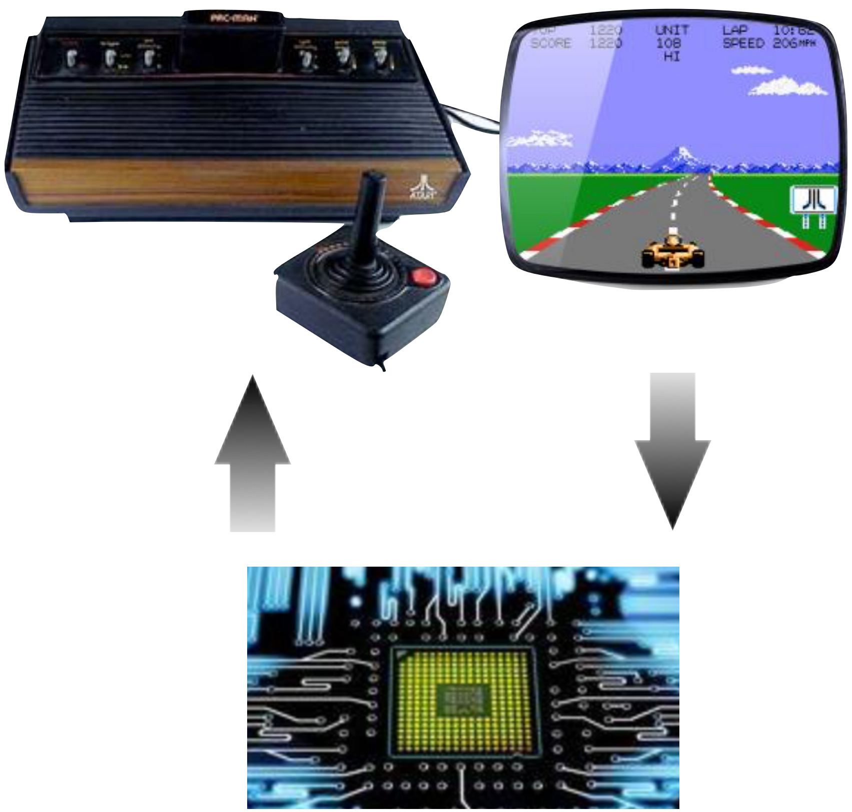
南京大学 人工智能学院
School of Artificial Intelligence, Nanjing University

Lecture 8
Deep Reinforcement Learning & Advanced Q-Learning

The Atari games

Deepmind Deep Q-learning on Atari

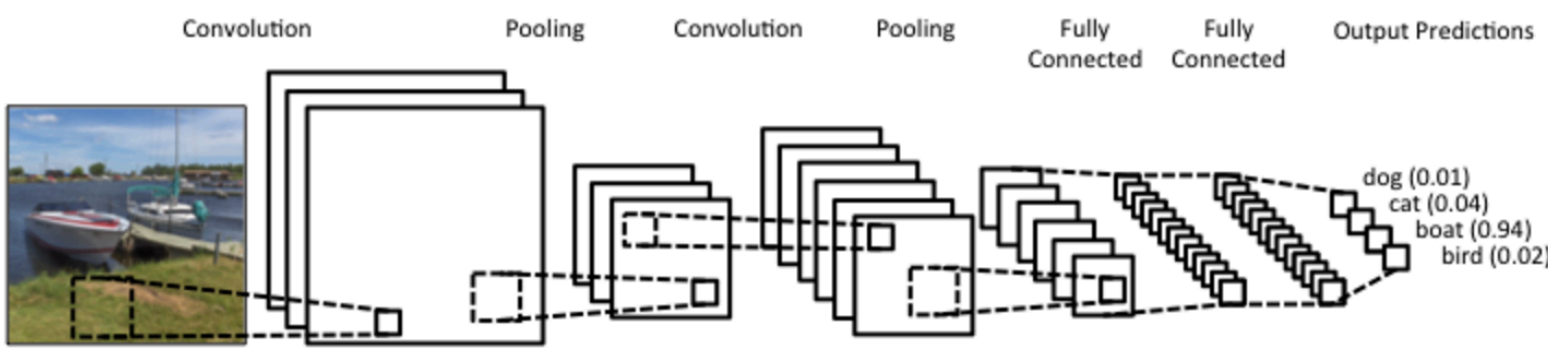
[Mnih *et al*. Human-level control through deep reinforcement learning. Nature, 518(7540): 529-533, 2015]



Eye of agent: Deep learning

a powerful architecture for image analysis

- differentiable
- require a lot of samples to train



Deep reinforcement learning

= deep model + reinforcement learning:

deep model as the function approximation / policy model

How to fit deep neural networks?

- stability?
- data?
- network structure?
- …

Deep Q-Network

DQN

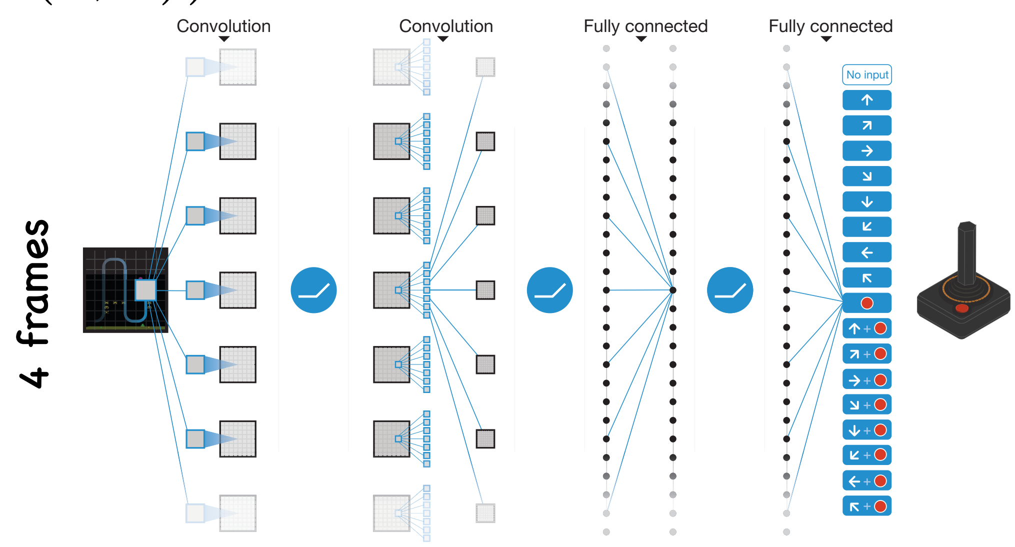
[Mnih *et al*. Human-level control through deep reinforcement learning. Nature, 518(7540): 529-533, 2015]

- using ϵ-greedy policy
- store 1 million recent history (s,a,r,s’) in replay memory D
- sample a mini-batch (32) from D
- calculate Q-learning target $\tilde{Q}$
- update CNN by minimizing the Bellman error (delayed update)

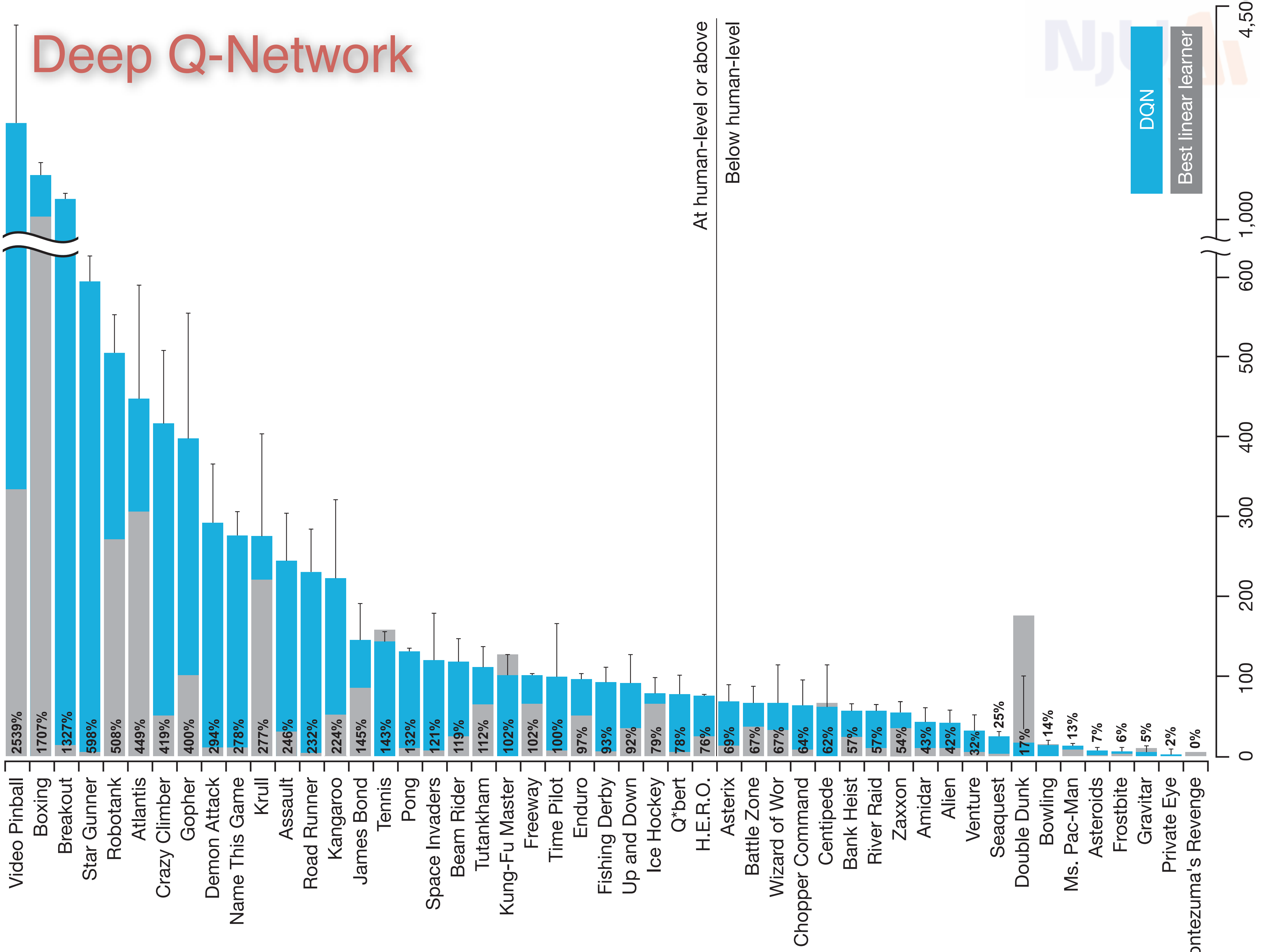
$$\sum (r + \gamma \max_{a'} \tilde{Q}(s', a') - Q_w(s, a))^2$$

DQN on Atari

learn to play from pixels



Deep Q-Network



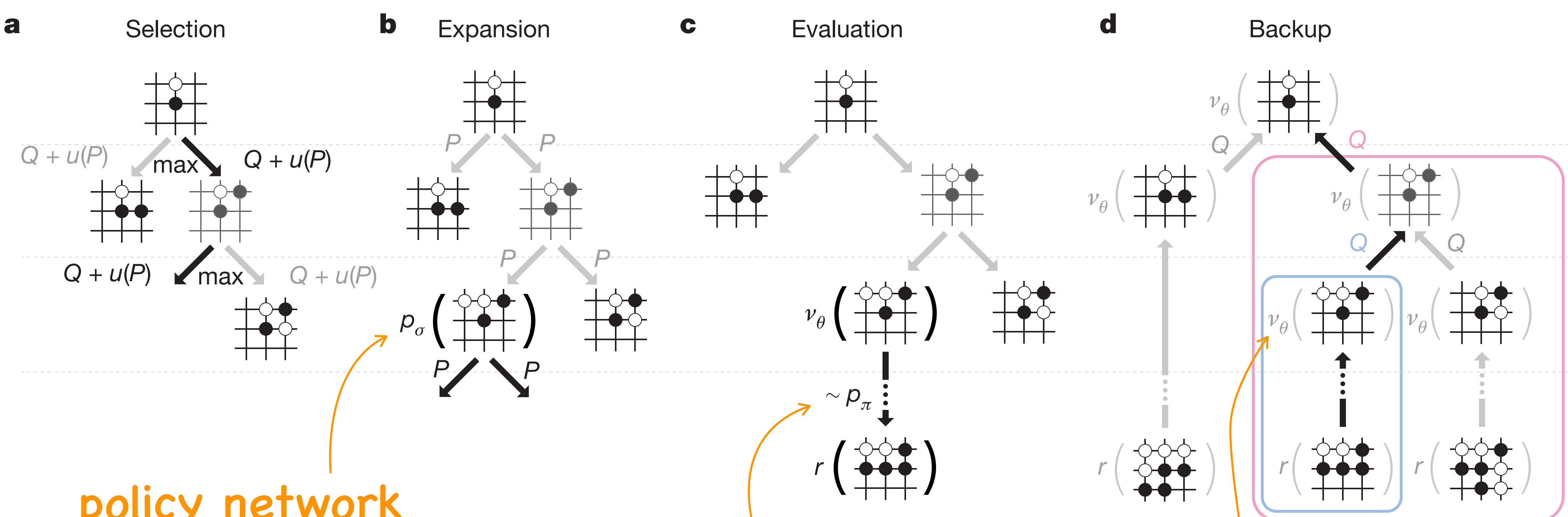
Deep Q-Network

effectiveness

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

AlphaGo

A combination of tree search, deep neural networks and reinforcement learning



policy network

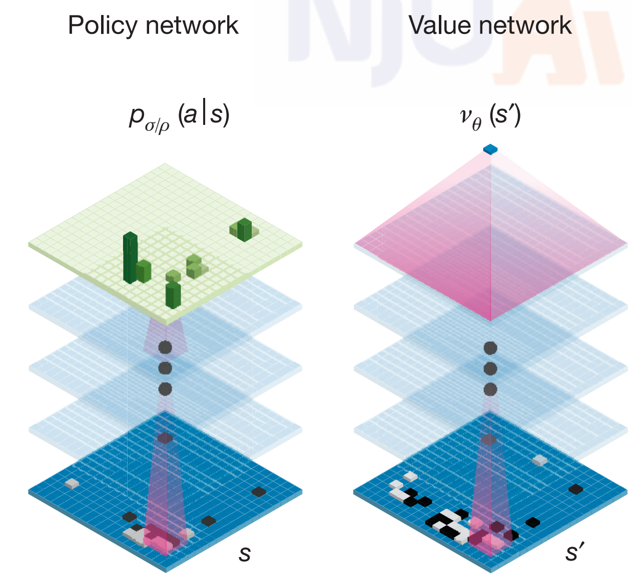
fast roll-out policy

value network

AlphaGo

policy network: a CNN output π(s,a)

value network: a CNN output V(s)



Feature	# of planes	Description
Stone colour	3	Player stone / opponent stone / empty
Ones	1	A constant plane filled with 1
Turns since	8	How many turns since a move was played
Liberties	8	Number of liberties (empty adjacent points)
Capture size	8	How many opponent stones would be captured
Self-atari size	8	How many of own stones would be captured
Liberties after move	8	Number of liberties after this move is played
Ladder capture	1	Whether a move at this point is a successful ladder capture
Ladder escape	1	Whether a move at this point is a successful ladder escape
Sensibleness	1	Whether a move is legal and does not fill its own eyes
Zeros	1	A constant plane filled with 0
Player color	1	Whether current player is black

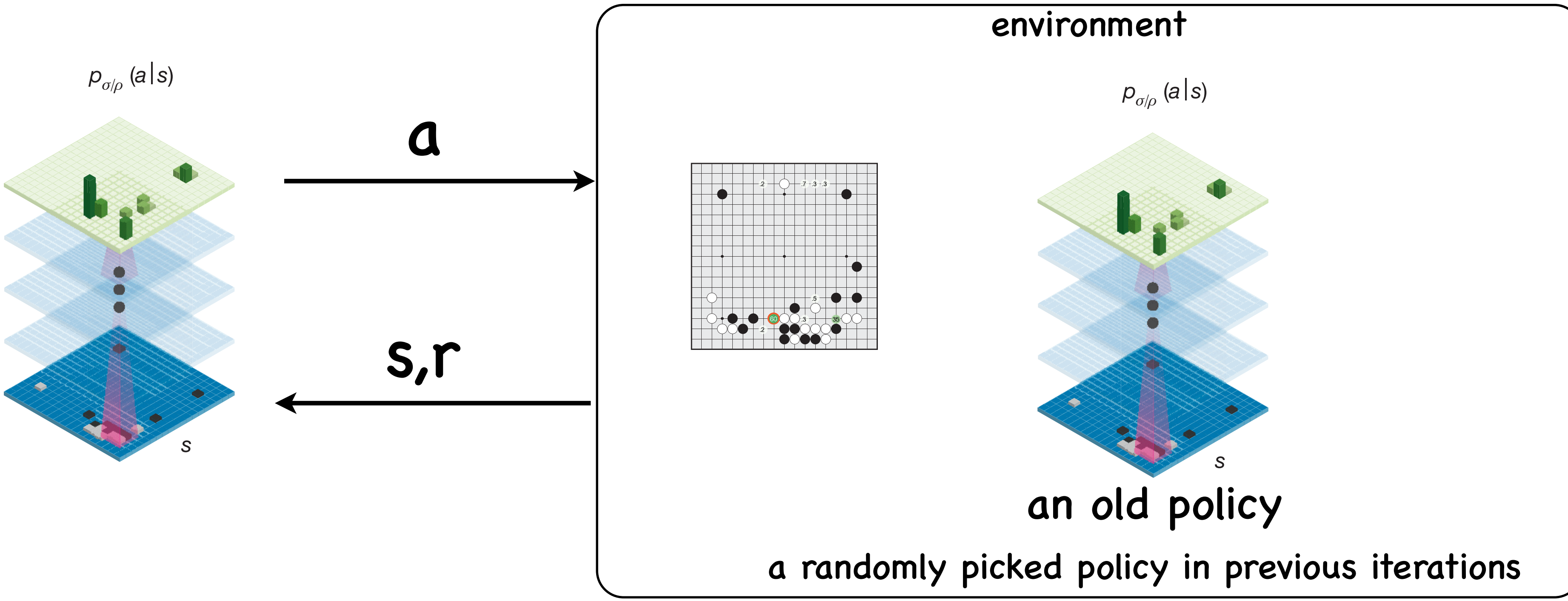
policy network: initialization

supervised learning from human v.s. human data

Architecture			Evaluation				
Filters	Symmetries	Features	Test accuracy %	Train accuracy %	Raw net wins %	*AlphaGo* wins %	Forward time (ms)
128	1	48	54.6	57.0	36	53	2.8
192	1	48	55.4	58.0	50	50	4.8
256	1	48	55.9	59.1	67	55	7.1
256	2	48	56.5	59.8	67	38	13.9
256	4	48	56.9	60.2	69	14	27.6
256	8	48	57.0	60.4	69	5	55.3
192	1	4	47.6	51.4	25	15	4.8
192	1	12	54.7	57.1	30	34	4.8
192	1	20	54.7	57.2	38	40	4.8
192	8	4	49.2	53.2	24	2	36.8
192	8	12	55.7	58.3	32	3	36.8
192	8	20	55.8	58.4	42	3	36.8

policy network: further improvement

reinforcement learning



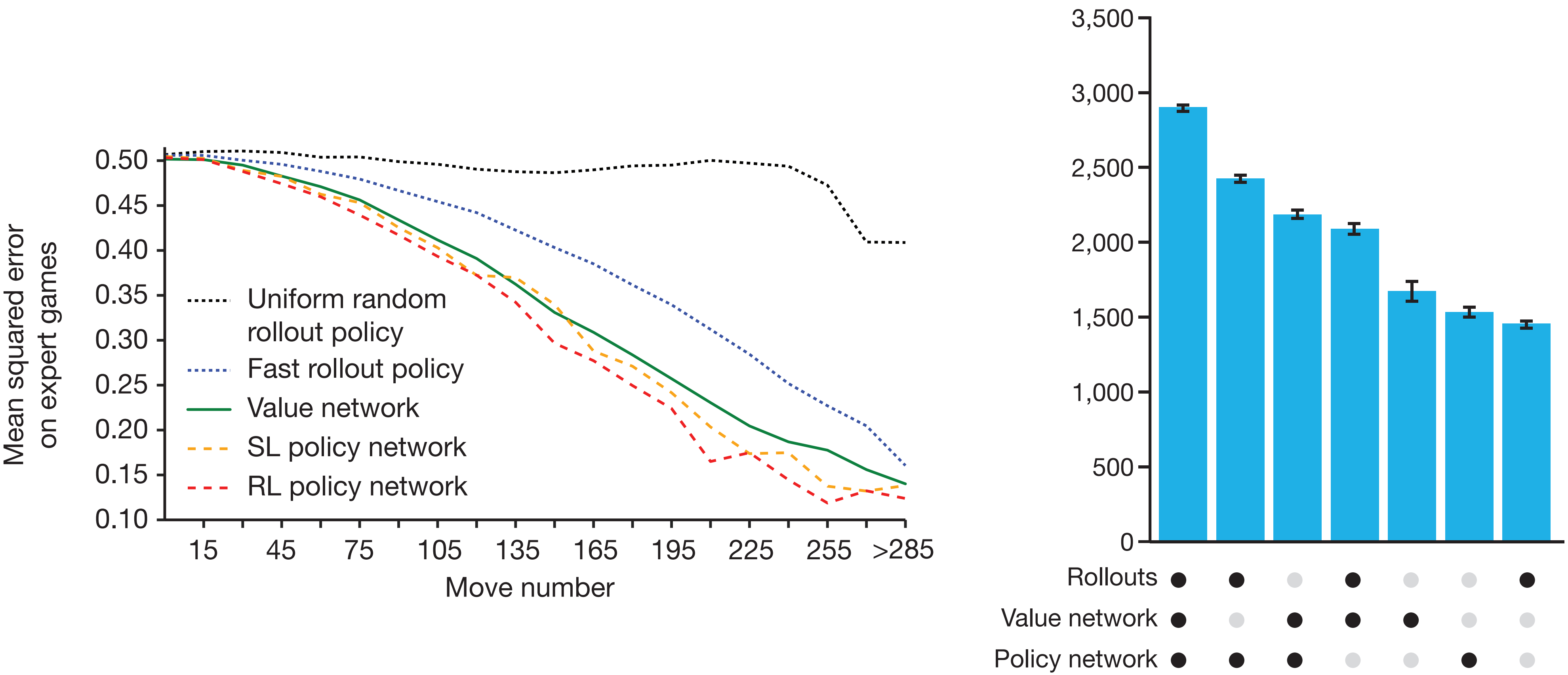
a.k.a. self-play

reward:

+1 -- win at terminate state

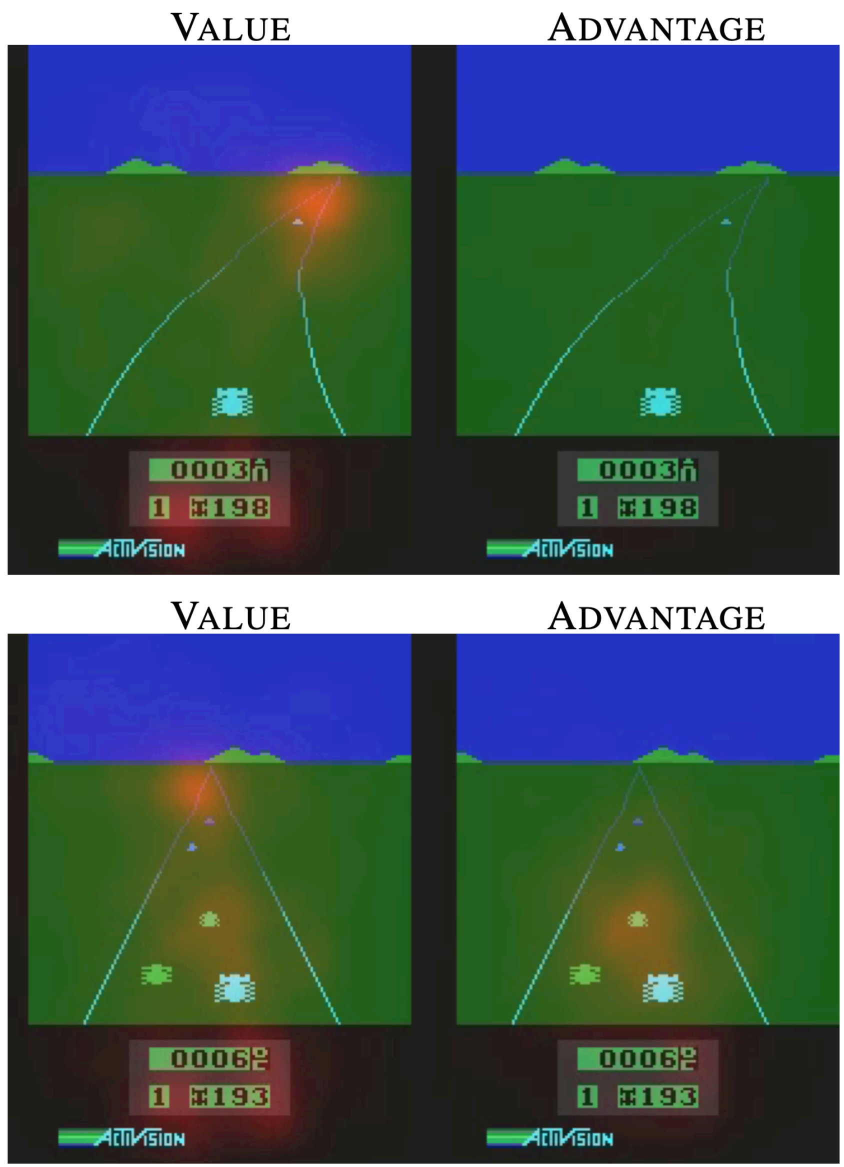
-1 -- loss at terminate state

value network: supervised learning from RL data

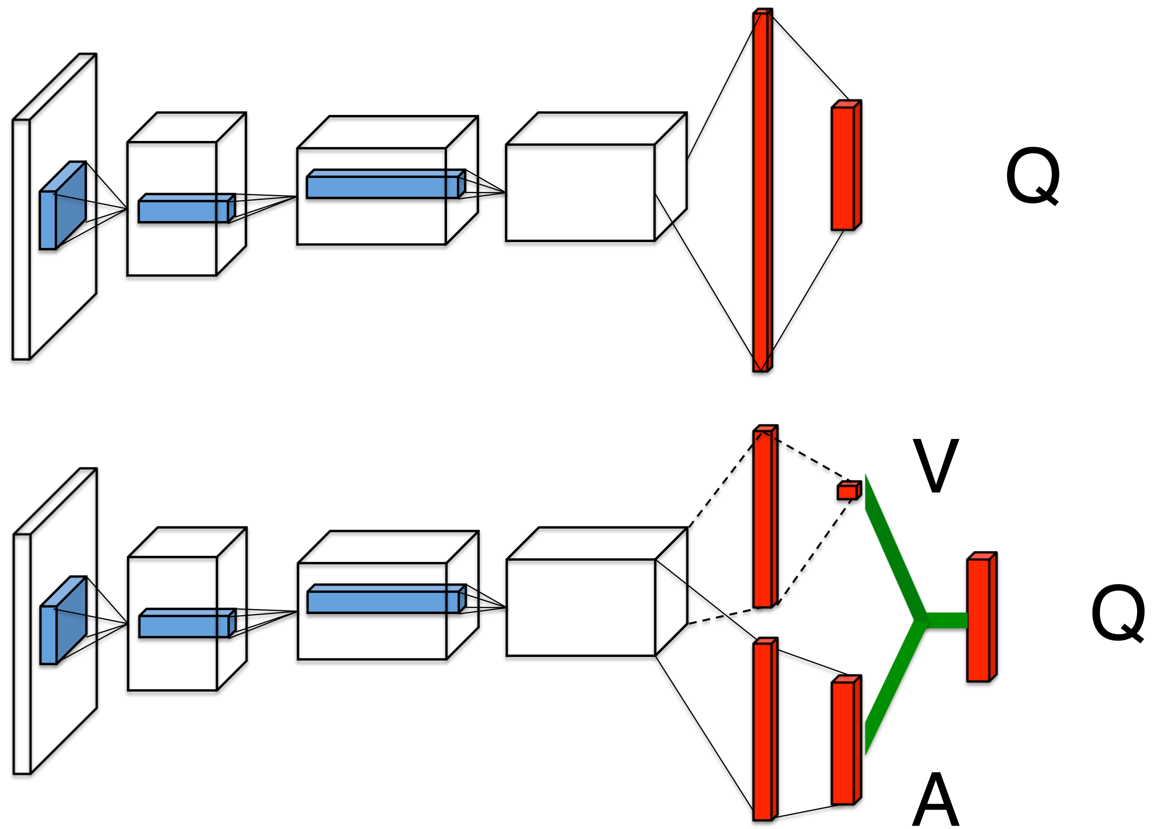


Dueling network architecture

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + A(s, a; \theta, \alpha)$$



Dueling network architecture

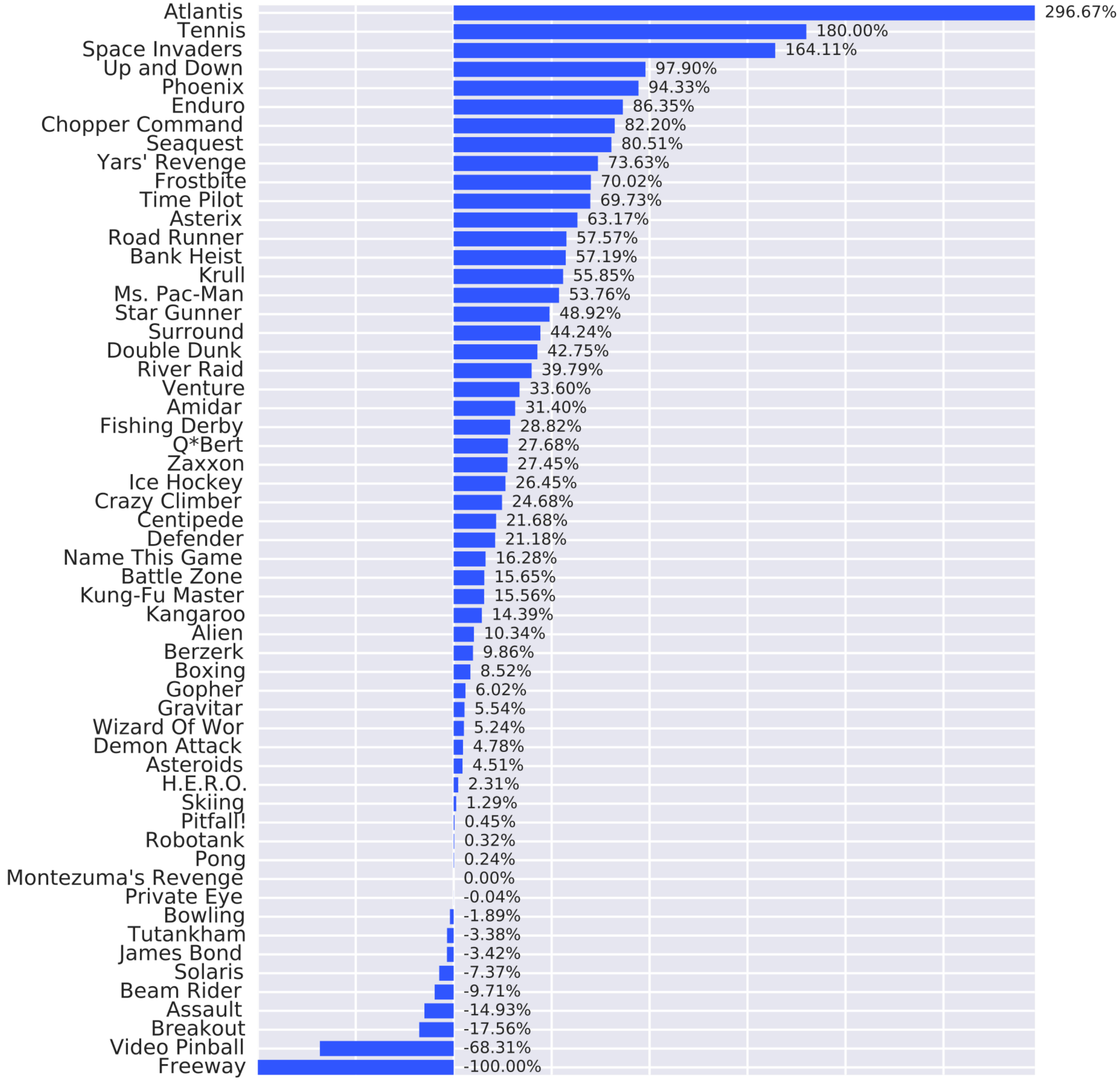


two versions of multi-solutions elimination:

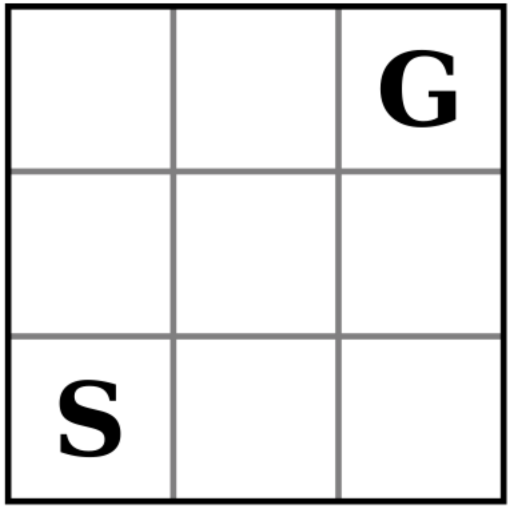
$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \max_{a' \in |\mathcal{A}|} A(s,a';\theta,\alpha)\right)$$

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s,a';\theta,\alpha)\right)$$

Dueling network architecture



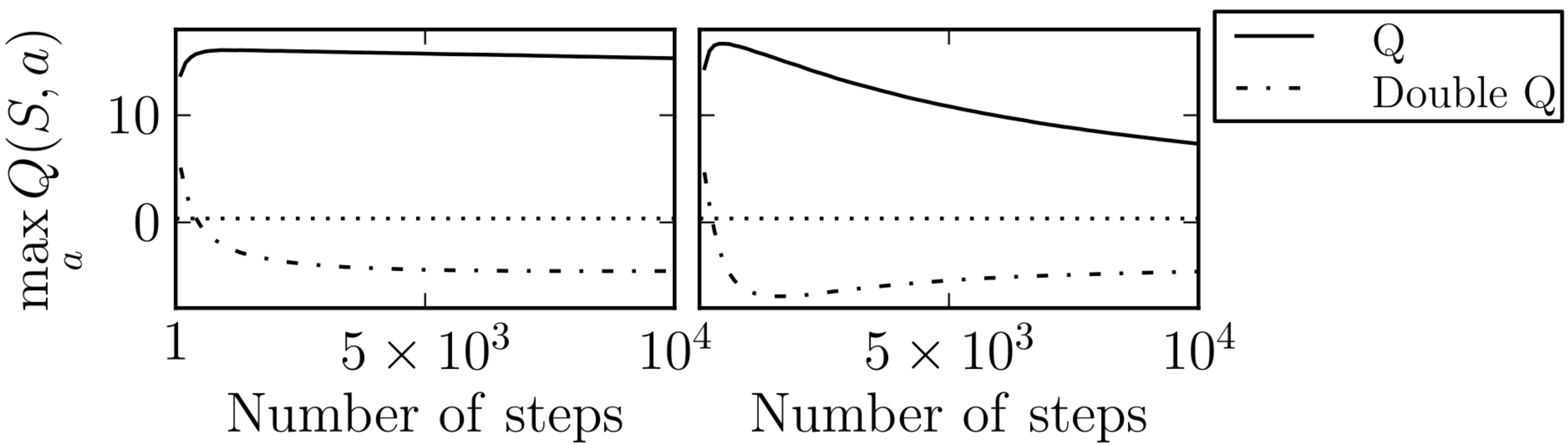
The overestimation problem of Q-learning



5 steps

reward:

- non-goal: -12 or +10 randomly
- goal: +5



The overestimation problem of Q-learning

ideally: $$Q^\pi(s,a) = E[\sum_{t=1}^{T} r_t | s, a]$$

$Q_0 = 0$, initial state
for i=0, 1, ...
$\quad a = \pi_\epsilon(s)$
$\quad s', r =$ do action a
$\quad a' = \pi(s')$
$\quad Q(s,a) \mathrel{+}= \alpha(r + \gamma Q(s', a') - Q(s,a))$
$\quad \pi(s) = \arg\max_a Q(s,a)$
$\quad s = s'$
end for

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t(s_t, a_t)\left(r_t + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)\right)$$

Single estimator

$$\mu_i(D) = \frac{1}{|D_i|} \sum_{d \in D_i} d$$

Example: 2-arm bandit

$|D| = 2000, |D_1| = 1000, |D_2| = 1000, X_1, X_2 \sim \mathcal{N}(0, 1)$

- $E\{X_1\} = 0, E\{X_2\} = 0 \Rightarrow \max_i E\{X_i\} = 0$
- If $\mu_1(D) = 0.01, \mu_2(D) = -0.01$, then $\max_i \mu_i(D) = 0.01 > 0$

average is an unbiased estimator of expectation
but max is not

$$E\{\max_i \mu_i(D)\} \geq \max_i E\{X_i\}$$

Double estimator

- **Double estimator** divides the sample set D into two disjoint subsets, D^U and D^V
- It uses

$$\mu_{a^*}^V(D)$$

to estimate $\max_i E\{X_i\}$, where $a^* \in \arg\max_i \mu_i^U(D)$

Example: 2-arm bandit

$|D| = 2000, |D_1| = 1000, |D_2| = 1000, X_1 \sim \mathcal{N}(0.01, 1), X_2 \sim \mathcal{N}(0, 1)$
$|D_1^U| = |D_1^V| = 500, |D_2^U| = |D_2^V| = 500$

- If $\mu_1^U(D) = -0.01, \mu_2^U(D) = 0$, then $\max_i \mu_i^U(D) = 0, a^* = 2$
- $E\{\mu_{a^*}^V(D)\} = E\{X_2\} < E\{X_1\}$

- $E\{\mu_{a^*}^V(D)\} \leq \max_i E\{X_i\}$ (**underestimation**)
  - Double estimator is **unbiased** when the variables are i.i.d.

Double DQN

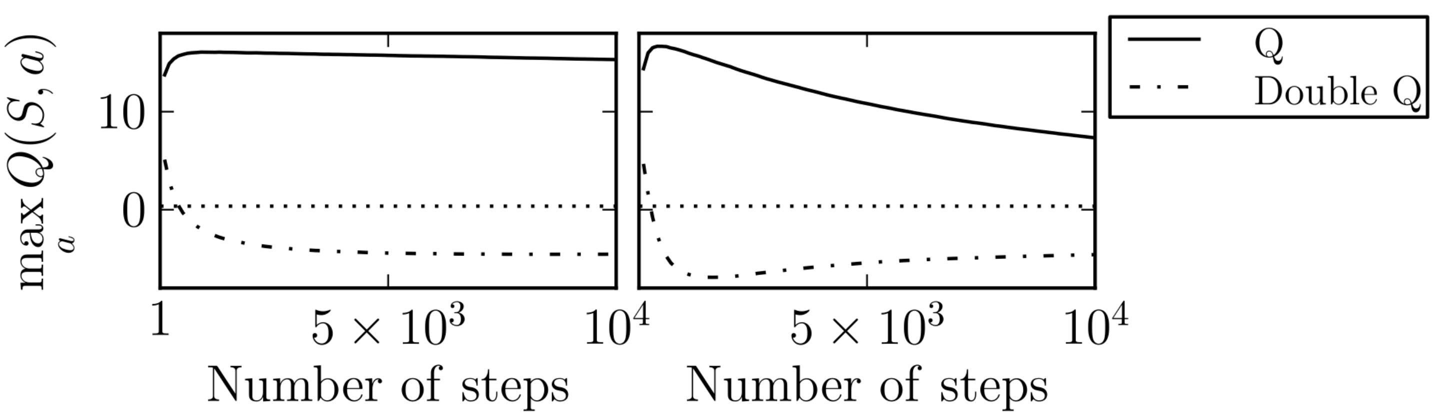
- It stores two Q functions, Q^U and Q^V, and uses two separate subsets of experience samples to learn them
- Update rule of Q^U:

$$Q^U(s,a) \leftarrow Q^U(s,a) + \alpha^U(s,a)[r + \gamma Q^V(s', a^*) - Q^U(s,a)]$$

where $a^* \leftarrow \arg\max_a Q^U(s', a)$

- Underestimation of action values:

$$E\{Q^V(s', a^*)\} = E\{Q(s', a^*)\} \leq \max_{a'} E\{Q(s', a')\}$$



Variants

Q-learning [Watkins, PhD Thesis 1989]
  Single estimator, overestimation

Double Q-learning [van Hasselt, NIPS 2010]
  Double estimator, underestimation

Weighted double Q-learning [Zhang et al., IJCAI 2017]
  Weighted double estimator, trade-off between overestimation and underestimation

$$Q^{U,WDE}(s', a^*) = \beta^U Q^U(s', a^*) + (1 - \beta^U) Q^V(s', a^*)$$

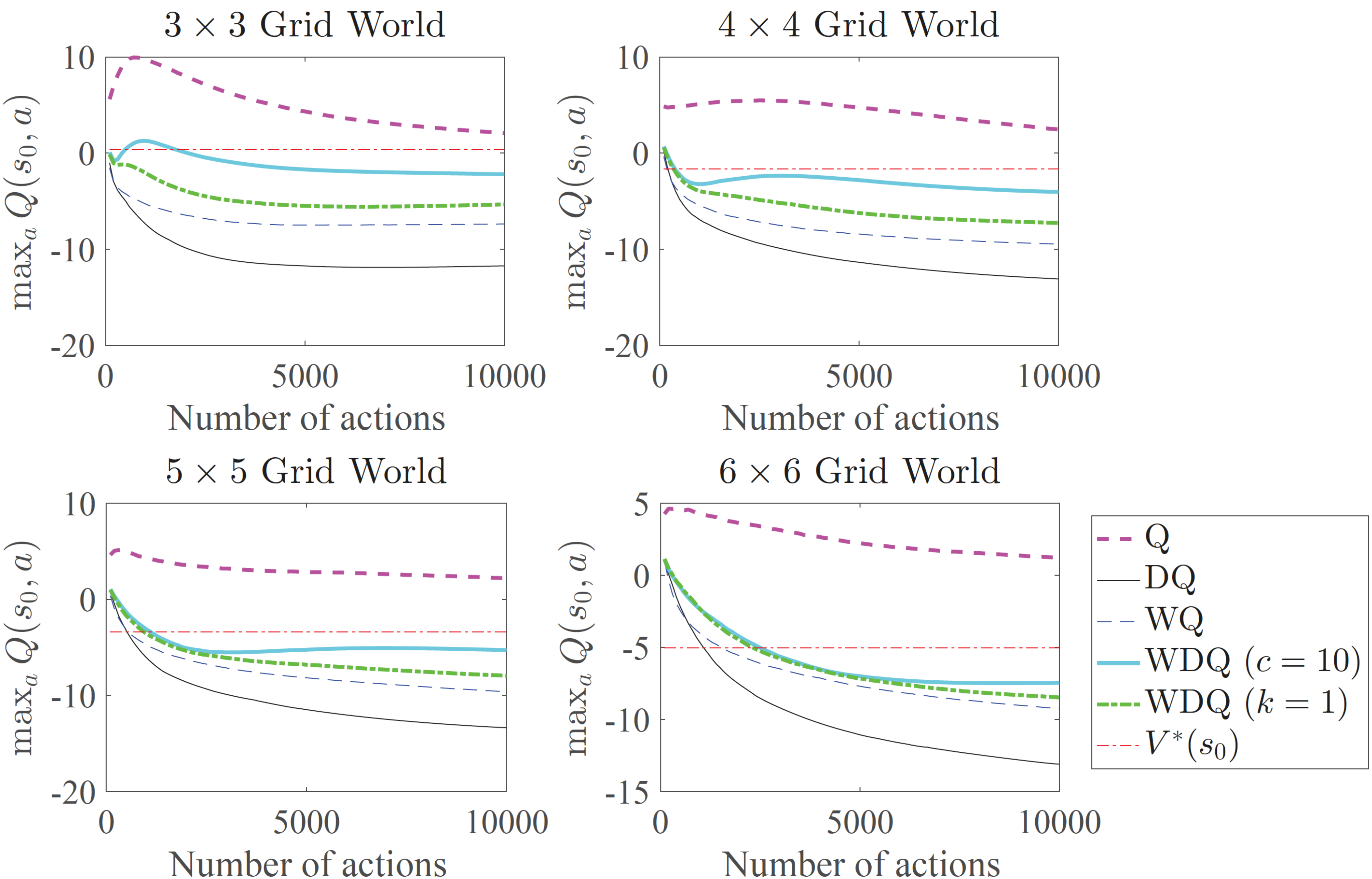
where

$$\beta^U \leftarrow \frac{|Q^V(s', a^*) - Q^V(s', a_L)|}{c + |Q^V(s', a^*) - Q^V(s', a_L)|}$$

Bias-corrected Q-learning [Lee and Powell, AAAI 2012]
  Single estimator - a bias correction term

Weighted Q-learning [D'Eramo et al., ICML 2016]
  Weighted estimator, a weighted mean of all the sample means (normal distributions)

Variants



3 × 3 Grid World
4 × 4 Grid World
5 × 5 Grid World
6 × 6 Grid World
$\max_a Q(s_0, a)$
Number of actions
10
0
-10
-20
5
-5
-15
5000
10000
Q
DQ
WQ
WDQ ($c = 10$)
WDQ ($k = 1$)
$V^*(s_0)$

A state-of-the-art: REDQ

Algorithm 1 Randomized Ensembled Double Q-learning (REDQ)

Initialize policy parameters θ, N Q-function parameters ϕ_i, $i = 1, \ldots, N$, empty replay buffer $\mathcal{D}$. Set target parameters $\phi_{\text{targ},i} \leftarrow \phi_i$, for $i = 1, 2, \ldots, N$

repeat

  Take one action $a_t \sim \pi_\theta(\cdot|s_t)$. Observe reward r_t, new state s_{t+1}.

  Add data to buffer: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, s_{t+1})\}$

  for G updates **do**

    Sample a mini-batch $B = \{(s, a, r, s')\}$ from $\mathcal{D}$

    Sample a set $\mathcal{M}$ of M distinct indices from $\{1, 2, \ldots, N\}$

    Compute the Q target y (same for all of the N Q-functions):

$$y = r + \gamma \left(\min_{i \in \mathcal{M}} Q_{\phi_{\text{targ},i}}\left(s', \tilde{a}'\right) - \alpha \log \pi_\theta\left(\tilde{a}' \mid s'\right) \right), \quad \tilde{a}' \sim \pi_\theta\left(\cdot \mid s'\right)$$

    for $i = 1, \ldots, N$ **do**

      Update ϕ_i with gradient descent using

$$\nabla_\phi \frac{1}{|B|} \sum_{(s,a,r,s') \in B} \left(Q_{\phi_i}(s, a) - y\right)^2$$

      Update target networks with $\phi_{\text{targ},i} \leftarrow \rho \phi_{\text{targ},i} + (1 - \rho)\phi_i$

  Update policy parameters θ with gradient ascent using

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} \left(\frac{1}{N} \sum_{i=1}^{N} Q_{\phi_i}\left(s, \tilde{a}_\theta(s)\right) - \alpha \log \pi_\theta\left(\tilde{a}_\theta(s) | s\right) \right), \quad \tilde{a}_\theta(s) \sim \pi_\theta(\cdot \mid s)$$

[Xinyue Chen, Che Wang, Zijian Zhou, Keith Ross. Randomized Ensembled Double Q-Learning: Learning Fast without a Model. ICLR 2021]

Prioritized experience replay

In DQN, replay memory is key to stabilize training

many transitions have small errors, a few have large errors

D: (s,a,r,s')

Prioritizing with TD-error

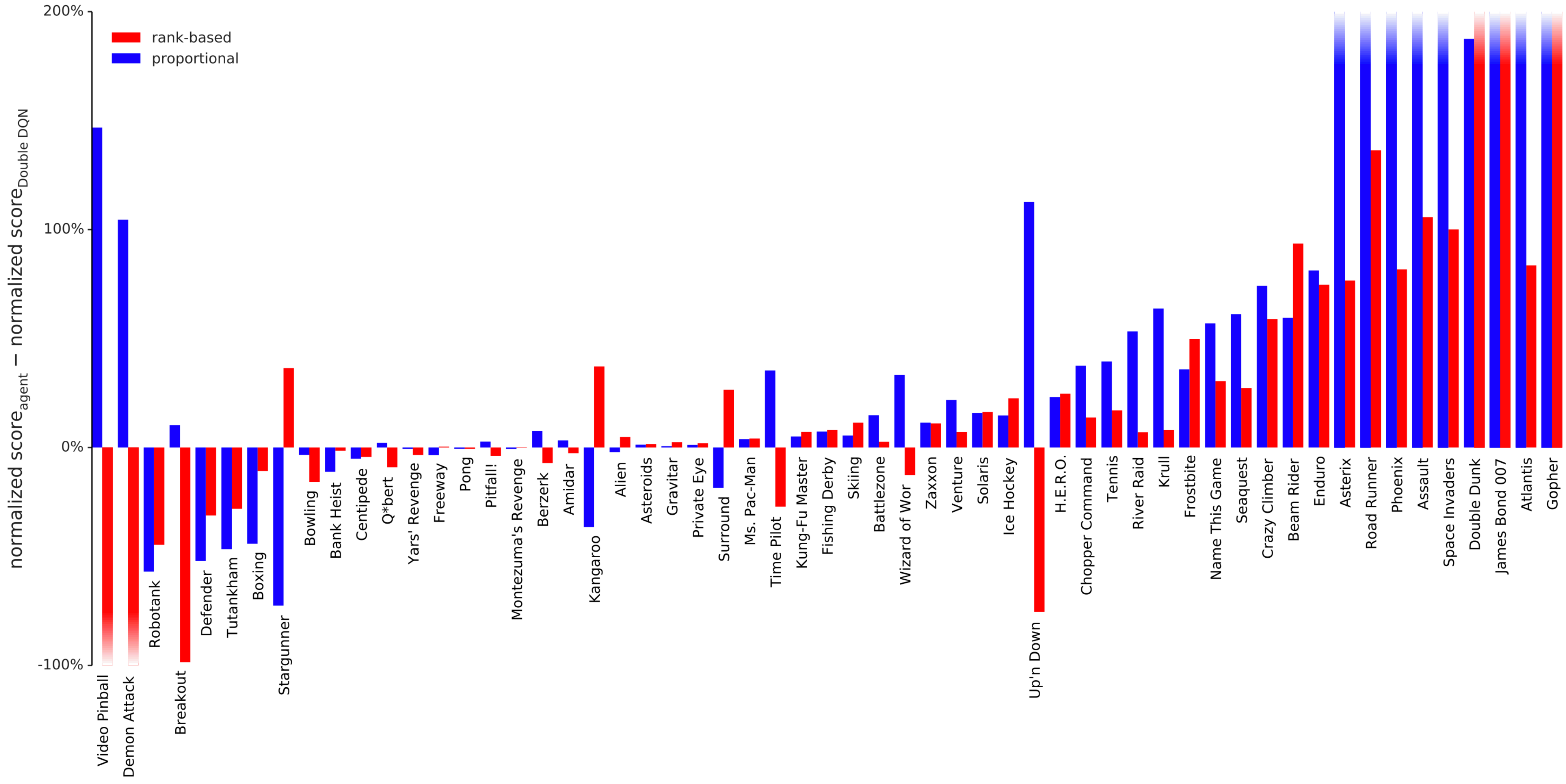
$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

where p_i is the TD-error or its variants on sample i

Annealing weight on TD-error:

$$w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)}\right)^\beta$$ with beta approaching 1 from 0

Prioritized experience replay



A solution to the prioritized exp. replay

$$\min_{w_k} \quad \eta(\pi^*) - \eta(\pi_k)$$

$$\text{s.t.} \quad Q_k = \operatorname*{argmin}_{Q \in \mathcal{Q}} \mathbb{E}_\mu[w_k(s,a) \cdot (Q - \mathcal{B}^* Q_{k-1})^2(s,a)]$$

$$w_k(s,a) = \underbrace{\frac{1}{Z^*}}_{(a)} \Bigg(\underbrace{\frac{d^{\pi_k}(s,a)}{\mu(s,a)}}_{(b)} \underbrace{(2 - \pi_k(a|s))}_{(c)} \underbrace{\exp(-|Q_k - Q^*|(s,a))}_{(d)} \underbrace{|Q_k - \mathcal{B}^* Q_{k-1}|(s,a)}_{(e)} + \underbrace{\epsilon_k(s,a)}_{(f)} \Bigg)$$

- (a): Normalization term.
- (b): Importance sampling term [Sinha et al., 2020].
- (c): Less action probability.
- (d): Closer value estimation to oracle [Kumar et al., 2020].
- (e): Higher hindsight Bellman error [Kumar et al., 2020].
- (f): Error term.

[Xuhui Liu, et al. Regret Minimization Experience Replay in Off-Policy Reinforcement Learning. NeurIPS 2021]