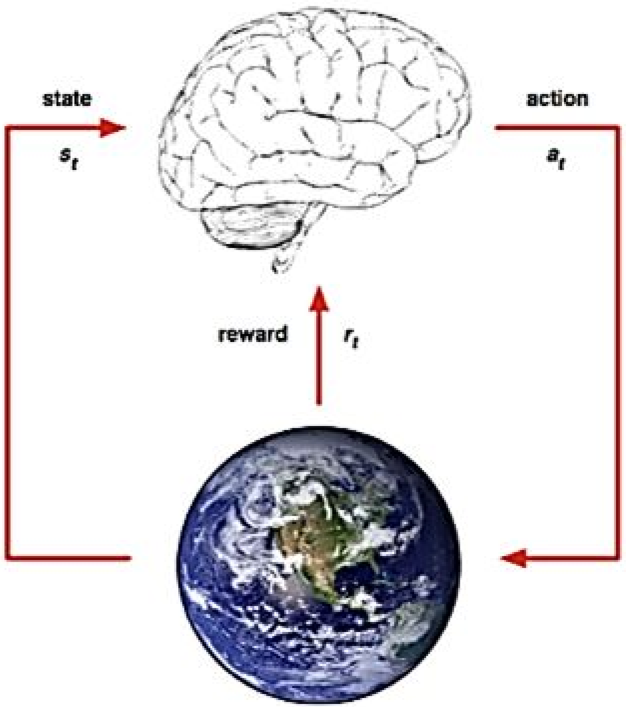
Нейросетевые методы в задачах обучения с подкреплением

Михаил Бурцев, к.ф.-м.н.,
лаб. “Нейронных систем и глубокого обучения”,
МФТИ

ОБУЧЕНИЕ С ПОДКРЕПЛЕНИЕМ

ПОСТАНОВКА ЗАДАЧИ



- На каждом шаге t агент:
  - получает состояние s_t
  - получает скалярное значение награды r_t
  - выполняет действие a_t
- Среда:
  - получает действие a_t
  - генерирует состояние s_t
  - генерирует скалярное значение награды r_t

ОСНОВНЫЕ ПОНЯТИЯ

- **Стратегия** π определяет выбор действия a для данного состояния среды s:

$$a = \pi(s)$$

- **Функция полезности** $Q^{\pi}(s, a)$ - полная ожидаемая награда при выборе действия a в состоянии s при использовании политики π :

$$Q^{\pi}(s, a) = \mathbb{E}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+2} + \ldots | s, a]$$

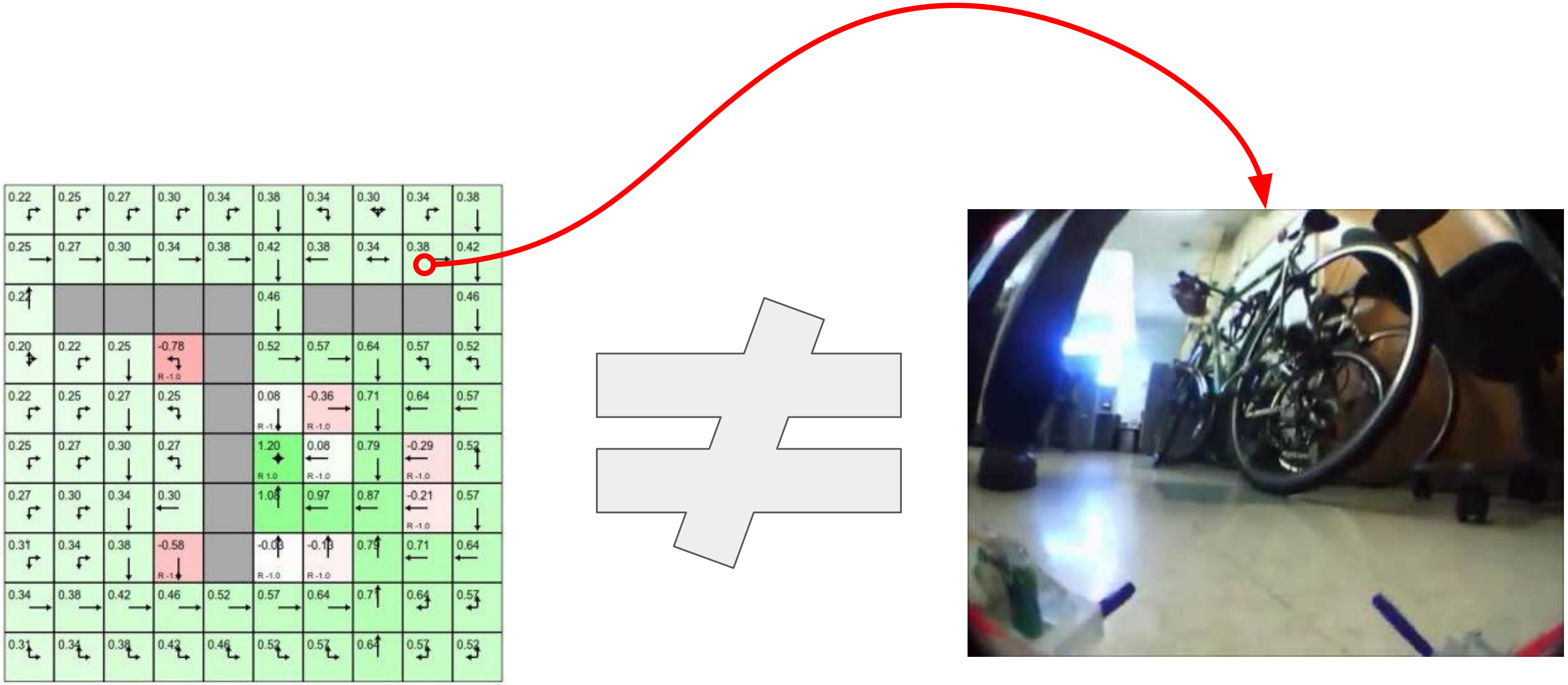
"Насколько хорошо действие a в состоянии s?"

ВЫЧИСЛЕНИЕ ПОЛЕЗНОСТИ

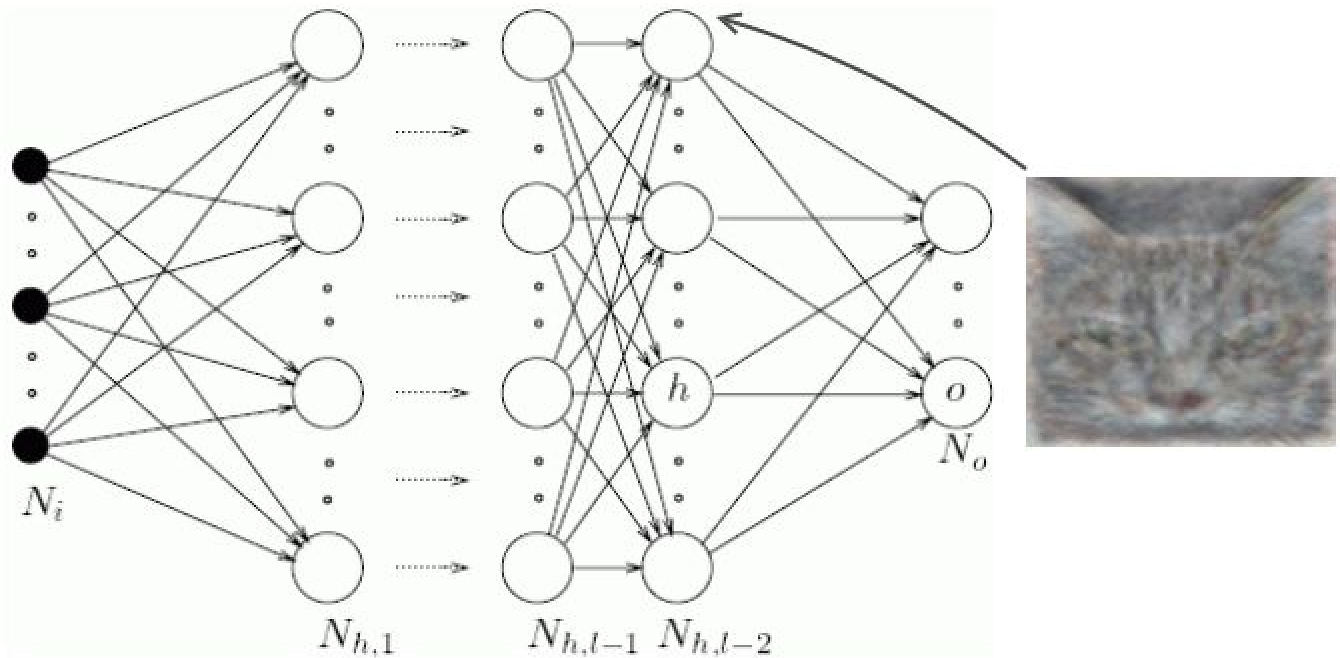
Решается итеративным расчетом функции полезности:

$$Q_{t+1}(s_t, a_t) = \underbrace{Q_t(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha_t(s_t, a_t)}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{R_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \underbrace{\max_a Q_t(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q_t(s_t, a_t)}_{\text{old value}} \right)$$

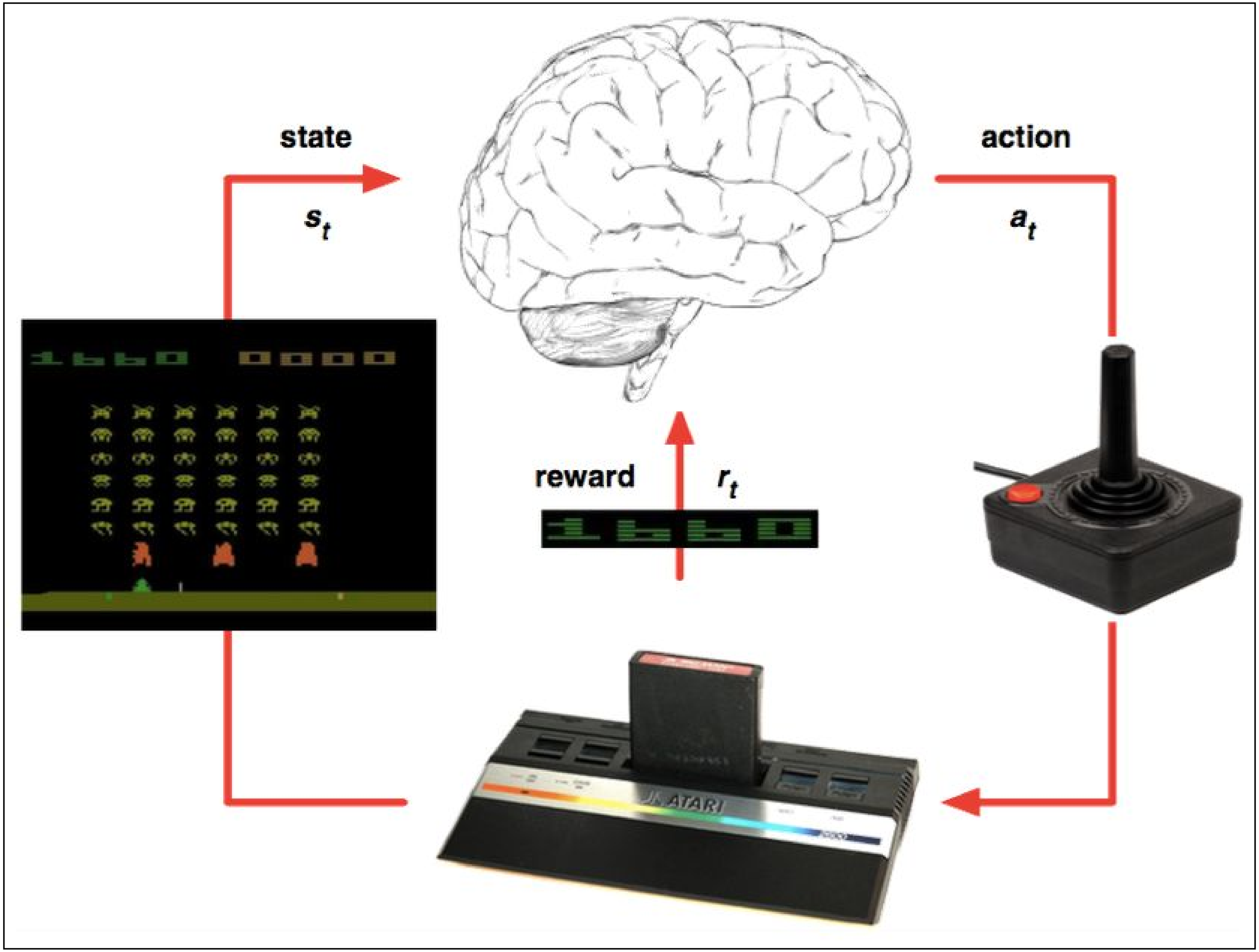
ПРОБЛЕМА “ПРОКЛЯТЬЯ РАЗМЕРНОСТИ”



НЕЙРОСЕТЬ, КАК УНИВЕРСАЛЬНЫЙ АППРОКСИМАТОР



ИГРА - МОДЕЛЬ РЕАЛЬНОСТИ


state
s_t
action
a_t
reward
r_t
ATARI

Playing Atari with Deep Reinforcement Learning

Volodymyr Mnih **Koray Kavukcuoglu** **David Silver** **Alex Graves** **Ioannis Antonoglou**

Daan Wierstra **Martin Riedmiller**

DeepMind Technologies

`{vlad,koray,david,alex.graves,ioannis,daan,martin.riedmiller} @ deepmind.com`

Abstract

We present the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning. The model is a convolutional neural network, trained with a variant of Q-learning, whose input is raw pixels and whose output is a value function estimating future rewards. We apply our method to seven Atari 2600 games from the Arcade Learning Environment, with no adjustment of the architecture or learning algorithm. We find that it outperforms all previous approaches on six of the games and surpasses a human expert on three of them.

Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." *arXiv preprint arXiv:1312.5602* (2013).

ГЛУБОКОЕ Q-ОБУЧЕНИЕ

- Представим функцию полезности глубокой Q-сетью (Q-network) с весами w:

$$Q(s, a, w) \approx Q^{\pi}(s, a)$$

- Определим целевую функцию как среднеквадратичную ошибку в значениях Q

$$\mathcal{L}(w) = \mathbb{E}[(r + \gamma \max_{a'} Q(s', a', w) - Q(s, q, w))^2]$$

Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." *arXiv preprint arXiv:1312.5602* (2013).

ПРОИГРЫВАНИЕ ОПЫТА

Для избавления от корреляций строим набор данных на основе собственного опыта агента

- Совершить действие a_t согласно ϵ-жадной стратегии
- Сохранить переход $(s_t, a_t, r_{t+1}, s_{t+1})$ в памяти для опыта $\mathcal{D}$
- Выбрать случайную мини-выборку переходов (s, a, r, s') из $\mathcal{D}$
- Уменьшаем СКО между Q-сетью и целевыми Q-значениями, как например

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

Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." *arXiv preprint arXiv:1312.5602* (2013).

ПРОБЛЕМА

что-то не сходится…

ЗАМОРОЗКА Q-СЕТИ

Для избавления от осцилляций фиксируем параметры в целевой Q-сети

- Вычисляем новые значения целевой Q-сети относительно фиксированных w^-

$$r + \gamma \max_{a'} Q(s', a', w^-)$$

- Уменьшаем СКО между Q-сетью и целевыми Q-значениями

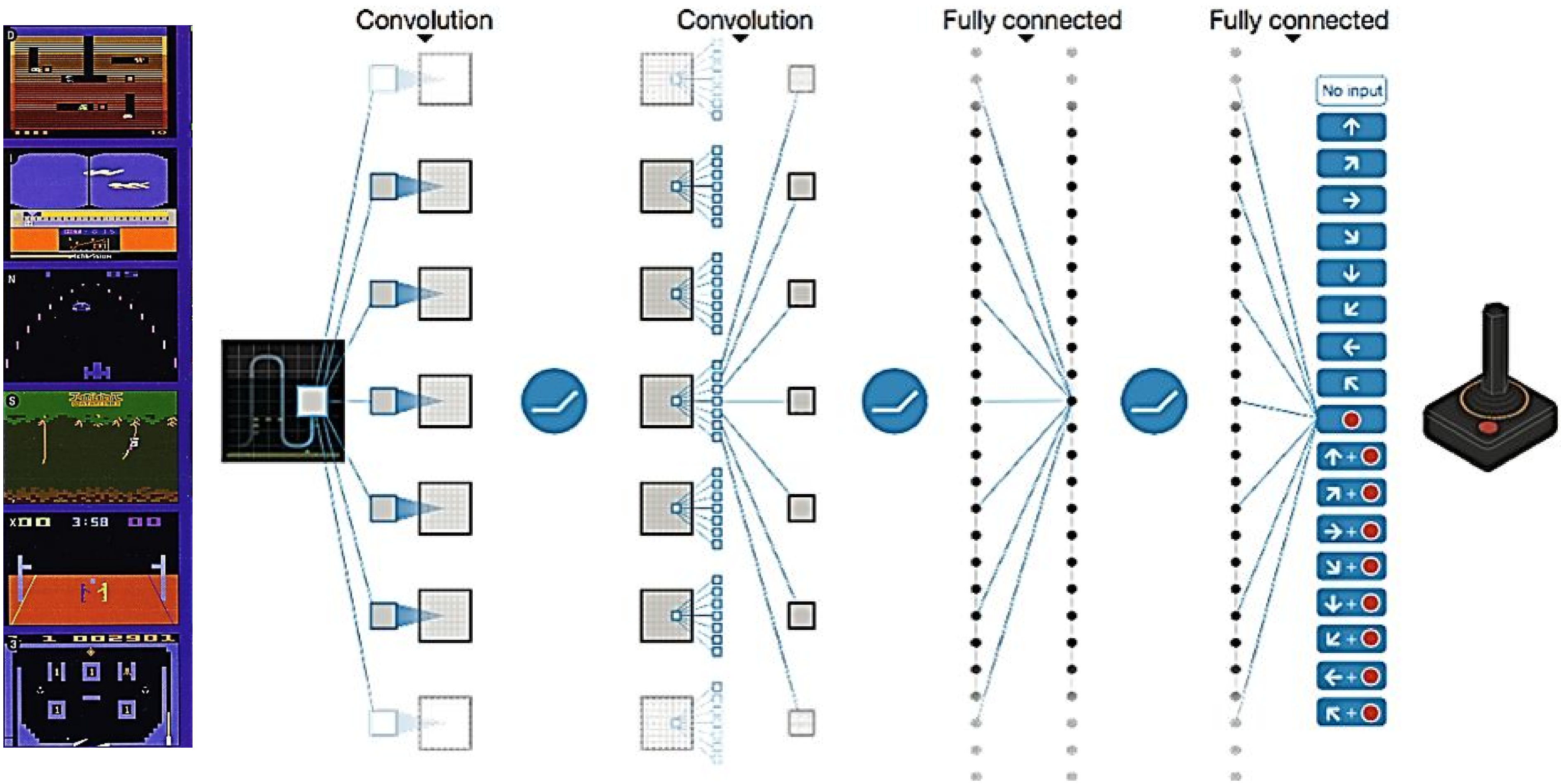
$$\mathcal{L} = \mathbb{E}_{s,a,r,s' \sim \mathbb{D}}[(r + \gamma \max_{a'} Q(s', a', w^-) - Q(s, a, w))^2]$$

- Время от времени обновляем зафиксированные параметры $w^- \leftarrow w$

ОГРАНИЧЕНИЕ АМПЛИТУДЫ НАГРАДЫ

- DQN ограничивает значения награды интервалом [-1,+1]
- Значения Q ограничены
- Обеспеспечивает хорошую обусловленность градиентов
- Не позволяет различить малые и большие значения награды

АРХИТЕКТУРА ГЛУБОКОЙ СЕТИ



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 $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." *arXiv preprint arXiv:1312.5602* (2013).

	B. Rider	Breakout	Enduro	Pong	Q*bert	Seaquest	S. Invaders
Random	354	1.2	0	−20.4	157	110	179
Sarsa [3]	996	5.2	129	−19	614	665	271
Contingency [4]	1743	6	159	−17	960	723	268
DQN	**4092**	**168**	**470**	**20**	**1952**	**1705**	**581**
Human	7456	31	368	−3	18900	28010	3690
HNeat Best [8]	3616	52	106	19	1800	920	**1720**
HNeat Pixel [8]	1332	4	91	−16	1325	800	1145
DQN Best	**5184**	**225**	**661**	**21**	**4500**	**1740**	1075

Table 1: The upper table compares average total reward for various learning methods by running an ϵ-greedy policy with $\epsilon = 0.05$ for a fixed number of steps. The lower table reports results of the single best performing episode for HNeat and DQN. HNeat produces deterministic policies that always get the same score while DQN used an ϵ-greedy policy with $\epsilon = 0.05$.

Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." *arXiv preprint arXiv:1312.5602* (2013).

19 Dec 2013

Playing Atari with Deep Reinforcement Learning

Volodymyr Mnih   Koray Kavukcuoglu   David Silver   Alex Graves   Ioannis Antonoglou

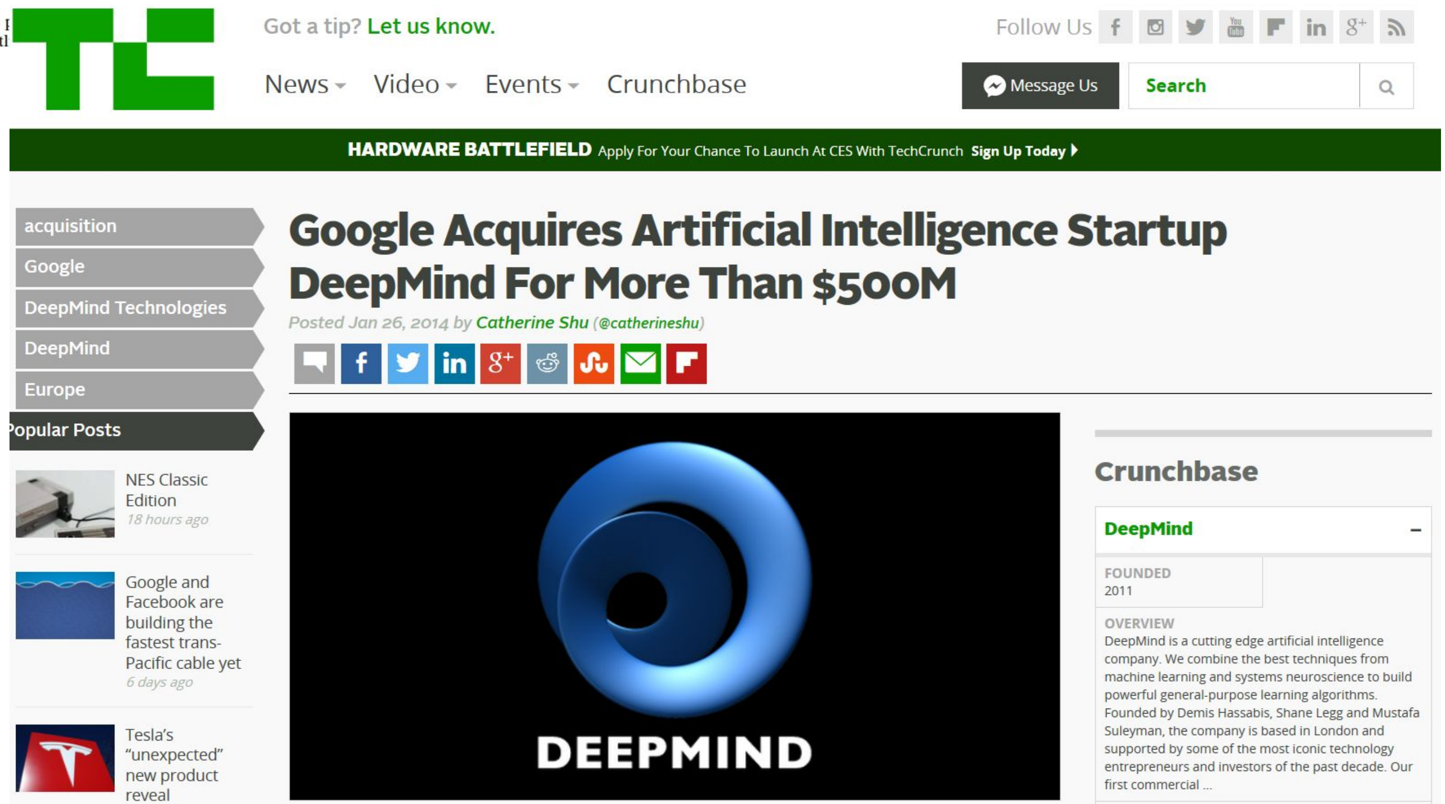
Daan Wierstra   Martin Riedmiller

DeepMind Technologies

{vlad,koray,david,alex.graves,ioannis,daan,martin.riedmiller} @ deepmind.com

Abstract

We p
rectl



Got a tip? Let us know.

Follow Us

News Video Events Crunchbase

Message Us

Search

HARDWARE BATTLEFIELD Apply For Your Chance To Launch At CES With TechCrunch Sign Up Today

acquisition
Google
DeepMind Technologies
DeepMind
Europe

Popular Posts

NES Classic Edition
18 hours ago

Google and Facebook are building the fastest trans-Pacific cable yet
6 days ago

Tesla's "unexpected" new product reveal

Google Acquires Artificial Intelligence Startup DeepMind For More Than $500M

Posted Jan 26, 2014 by Catherine Shu (@catherineshu)

DEEPMIND

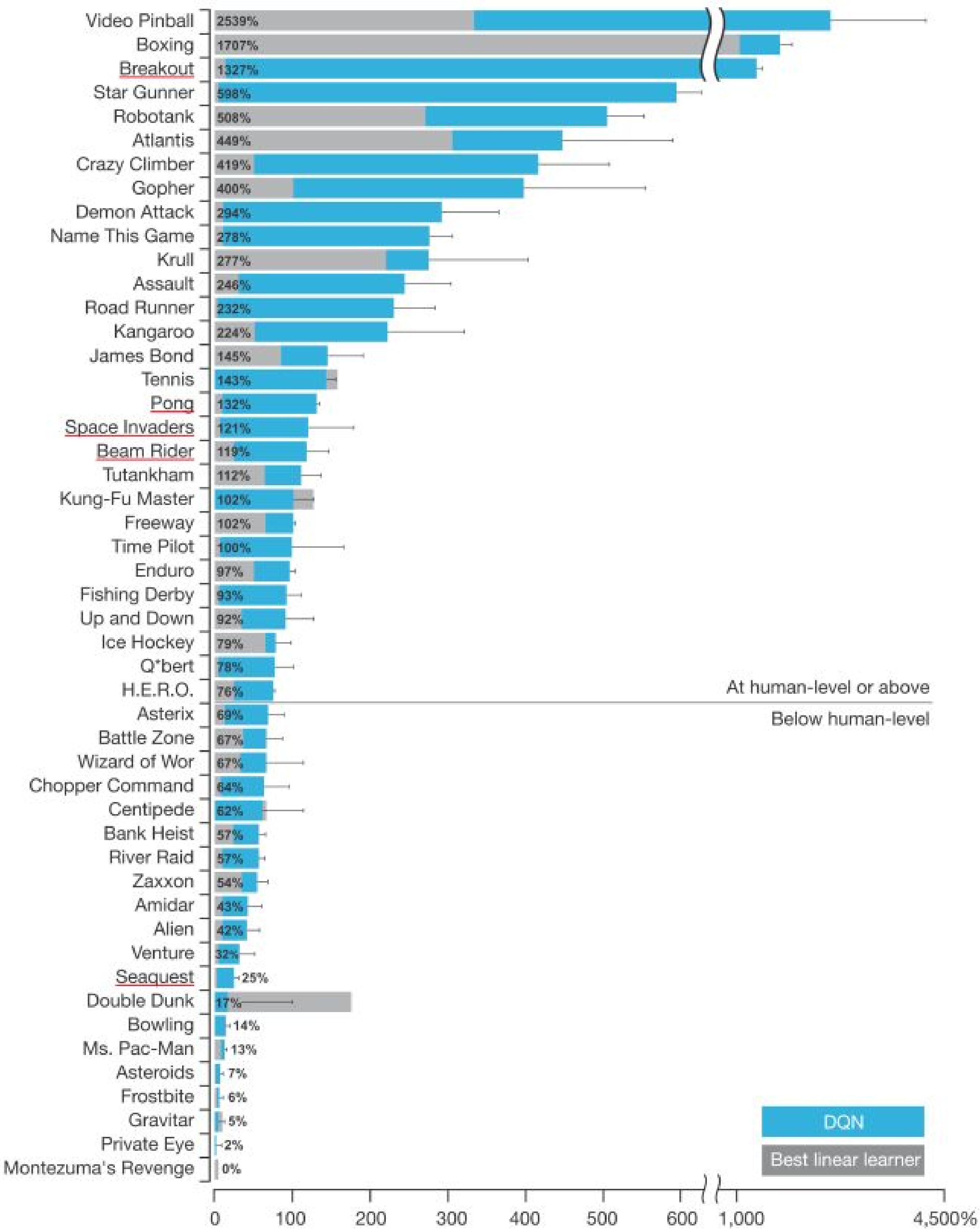
Crunchbase

DeepMind

FOUNDED
2011

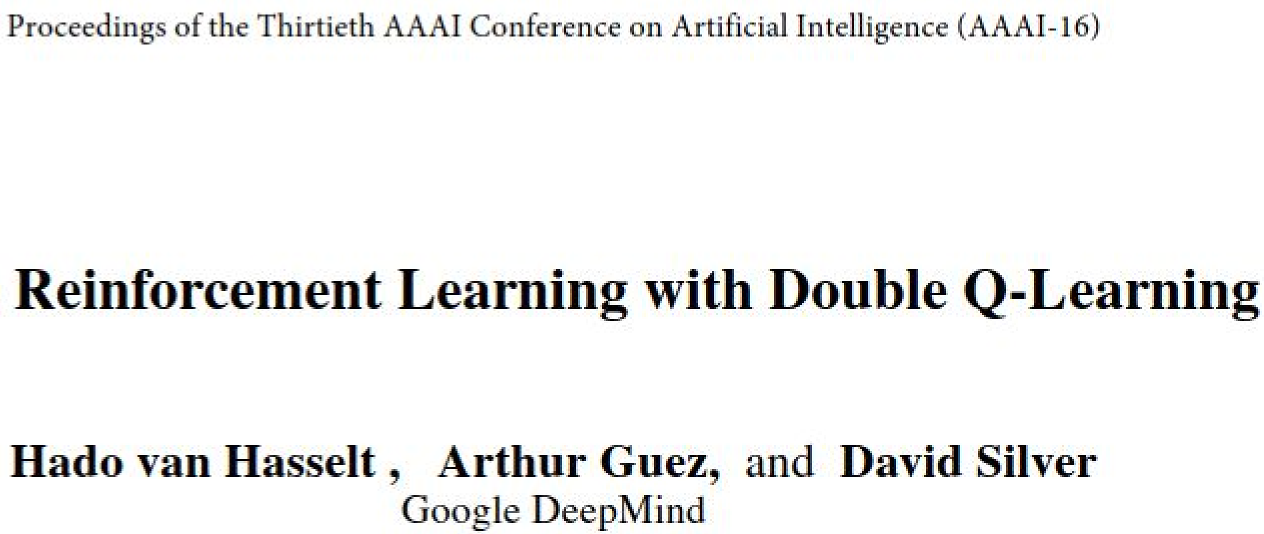
OVERVIEW
DeepMind is a cutting edge artificial intelligence company. We combine the best techniques from machine learning and systems neuroscience to build powerful general-purpose learning algorithms. Founded by Demis Hassabis, Shane Legg and Mustafa Suleyman, the company is based in London and supported by some of the most iconic technology entrepreneurs and investors of the past decade. Our first commercial ...


INNOVATIONS IN
The microbiome
nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
LEARNING CURVE
Self-taught AI software
attains human-level
performance in video games
PAGES 486 & 529
EPIDEMIOLOGY
SHARE DATA IN OUTBREAKS
Forge open access to sequences and more
PAGE 477
COSMOLOGY
A GIANT IN THE EARLY UNIVERSE
A supermassive black hole at a redshift of 6.3
PAGES 490 & 512
QUANTUM PHYSICS
TELEPORTATION FOR TWO
Transferring two properties of a single photon
PAGES 491 & 516
NATURE.COM/NATURE
26 February 2015
$10.00US $12.99CAN


Video Pinball
Boxing
Breakout
Star Gunner
Robotank
Atlantis
Crazy Climber
Gopher
Demon Attack
Name This Game
Krull
Assault
Road Runner
Kangaroo
James Bond
Tennis
Pong
Space Invaders
Beam Rider
Tutankham
Kung-Fu Master
Freeway
Time Pilot
Enduro
Fishing Derby
Up and Down
Ice Hockey
Q*bert
H.E.R.O.
Asterix
Battle Zone
Wizard of Wor
Chopper Command
Centipede
Bank Heist
River Raid
Zaxxon
Amidar
Alien
Venture
Seaquest
Double Dunk
Bowling
Ms. Pac-Man
Asteroids
Frostbite
Gravitar
Private Eye
Montezuma's Revenge
2539%
1707%
1327%
598%
508%
449%
419%
400%
294%
278%
277%
246%
232%
224%
145%
143%
132%
121%
119%
112%
102%
102%
100%
97%
93%
92%
79%
78%
76%
69%
67%
67%
64%
62%
57%
57%
54%
43%
42%
32%
25%
17%
14%
13%
7%
6%
5%
2%
0%
At human-level or above
Below human-level
DQN
Best linear learner
0
100
200
300
400
500
600
1,000
4,500%

SEAQUEST

Double DQN



Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep reinforcement learning with double Q-learning." *CoRR, abs/1509.06461* (2015).

Double DQN

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha(Y_t^{\mathrm{Q}} - Q(S_t, A_t; \boldsymbol{\theta}_t))\nabla_{\boldsymbol{\theta}_t} Q(S_t, A_t; \boldsymbol{\theta}_t)\,.$$

$$Y_t^{\mathrm{Q}} \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \boldsymbol{\theta}_t)\,.$$

$$Y_t^{\mathrm{Q}} = R_{t+1} + \gamma Q(S_{t+1}, \operatorname{argmax} Q(S_{t+1}, a; \boldsymbol{\theta}_t); \boldsymbol{\theta}_t)\,.$$

$$Y_t^{\mathrm{DoubleQ}} \equiv R_{t+1} + \gamma Q(S_{t+1}, \operatorname{argmax} Q(S_{t+1}, a; \boldsymbol{\theta}_t); \boldsymbol{\theta}'_t)\,.$$

$$Y_t^{\mathrm{DQN}} \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \boldsymbol{\theta}_t^-)\,.$$

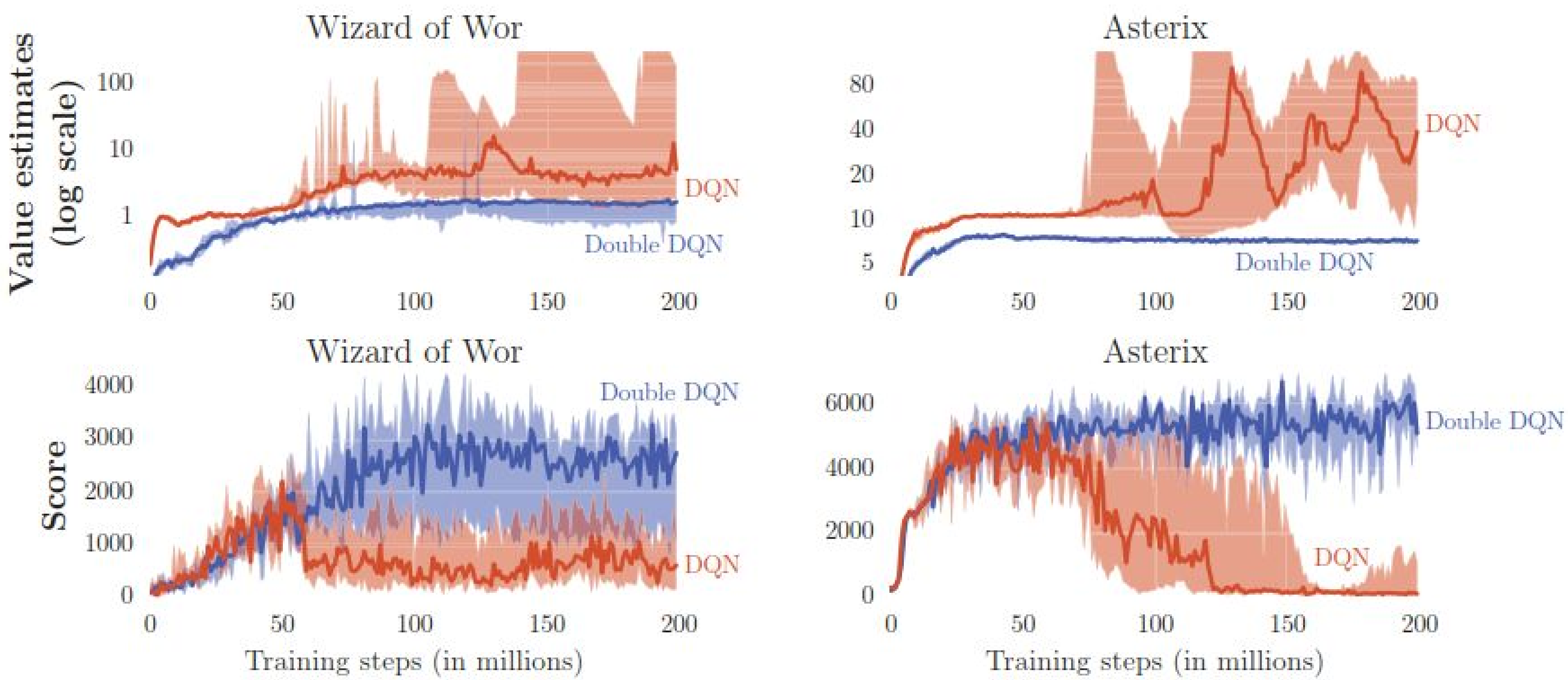
$$Y_t^{\mathrm{DoubleDQN}} \equiv R_{t+1} + \gamma Q(S_{t+1}, \operatorname*{argmax}_a Q(S_{t+1}, a; \boldsymbol{\theta}_t), \boldsymbol{\theta}_t^-)\,.$$

Algorithm 1: Double DQN Algorithm.

input : $\mathcal{D}$ – empty replay buffer; θ – initial network parameters, θ^- – copy of θ
input : N_r – replay buffer maximum size; N_b – training batch size; N^- – target network replacement freq.
for *episode* $e \in \{1, 2, \ldots, M\}$ **do**
  Initialize frame sequence $\mathbf{x} \leftarrow ()$
  for $t \in \{0, 1, \ldots\}$ **do**
    Set state $s \leftarrow \mathbf{x}$, sample action $a \sim \pi_{\mathcal{B}}$
    Sample next frame x^t from environment $\mathcal{E}$ given (s, a) and receive reward r, and append x^t to $\mathbf{x}$
    if $|\mathbf{x}| > N_f$ **then** delete oldest frame $x_{t_{min}}$ from $\mathbf{x}$ **end**
    Set $s' \leftarrow \mathbf{x}$, and add transition tuple (s, a, r, s') to $\mathcal{D}$,
      replacing the oldest tuple if $|\mathcal{D}| \geq N_r$
    Sample a minibatch of N_b tuples $(s, a, r, s') \sim \mathrm{Unif}(\mathcal{D})$
    Construct target values, one for each of the N_b tuples:
    Define $a^{\max}(s'; \theta) = \arg\max_{a'} Q(s', a'; \theta)$
    $$y_j = \begin{cases} r & \text{if } s' \text{ is terminal} \\ r + \gamma Q(s', a^{\max}(s'; \theta); \theta^-), & \text{otherwise.} \end{cases}$$
    Do a gradient descent step with loss $\|y_j - Q(s, a; \theta)\|^2$
    Replace target parameters $\theta^- \leftarrow \theta$ every N^- steps
  end
end

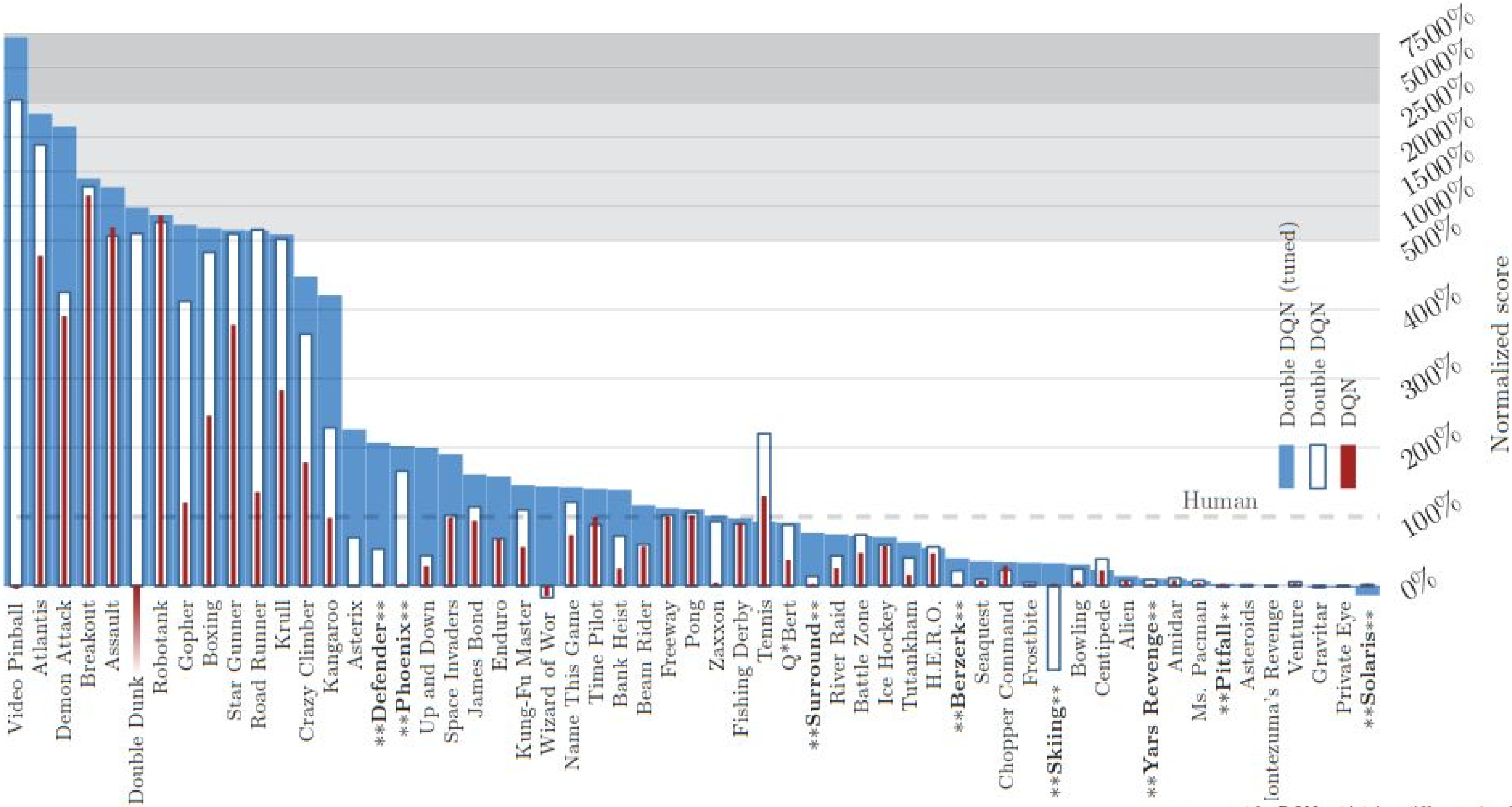
Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep reinforcement learning with double Q-learning." *CoRR, abs/1509.06461* (2015).

Double DQN



Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep reinforcement learning with double Q-learning." *CoRR, abs/1509.06461* (2015).

Double DQN



ters were tuned for DQN, which is a different algorithm. For the tuned version of Double DQN, we increased the number of frames between each two copies of the target network from 10,000 to 30,000, to reduce overestimations further because immediately after each switch DQN and Double DQN both revert to Q-learning. In addition, we reduced the exploration during learning from $\epsilon = 0.1$ to $\epsilon = 0.01$, and then used $\epsilon = 0.001$ during evaluation. Finally, the tuned version uses a single shared bias for all action values in the top layer of the network. Each of these changes improved performance and together they result in clearly better results.[3]

Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep reinforcement learning with double Q-learning." *CoRR, abs/1509.06461* (2015).

Prioritized Experience Replay

Tom Schaul, John Quan, Ioannis Antonoglou and David Silver
Google DeepMind
{schaul,johnquan,ioannisa,davidsilver}@google.com

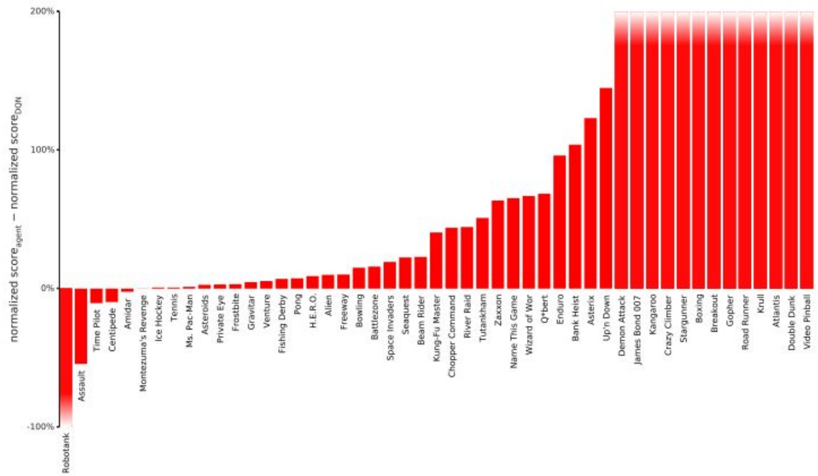


Figure 9: Difference in normalized score (the gap between random and human is 100%) on 49 games with human starts, comparing DQN with and without rank-based prioritized replay, showing substantial improvements in many games. Exact scores are in Table 6. See also Figure 3 where Double DQN is the baseline.

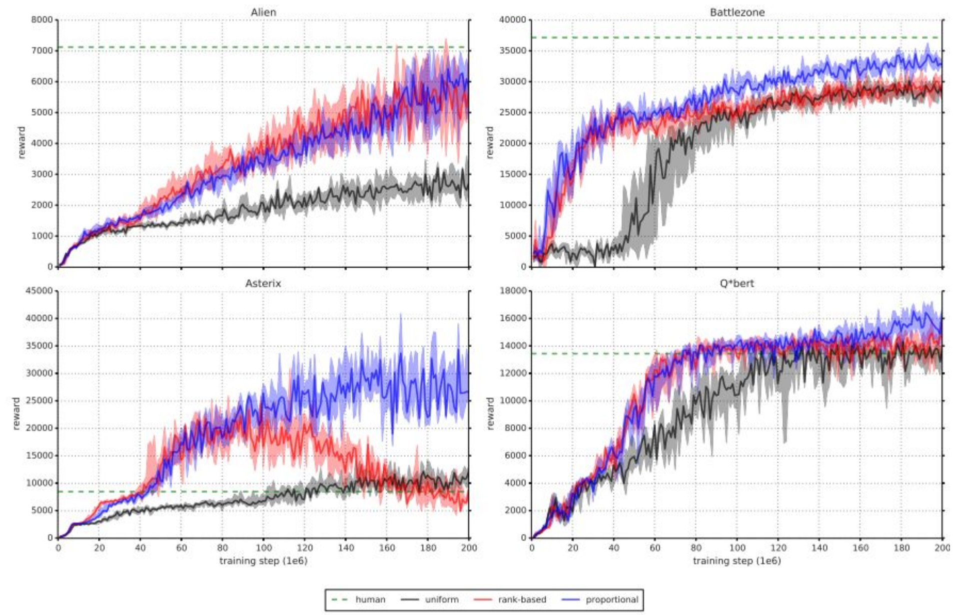


Figure 8: Detailed learning curves for rank-based (red) and proportional (blue) prioritization, as compared to the uniform Double DQN baseline (black) on a selection of games. The solid lines are the median scores, and the shaded area denotes the interquartile range across 8 random initializations. The dashed green lines are human scores. While the variability between runs is substantial, there are significant differences in final achieved score, and also in learning speed.

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha} \quad (1)$$

where $p_i > 0$ is the priority of transition i. The exponent α determines how much prioritization is used, with $\alpha = 0$ corresponding to the uniform case.

The first variant we consider is the direct, proportional prioritization where $p_i = |\delta_i| + \epsilon$, where ϵ is a small positive constant that prevents the edge-case of transitions not being revisited once their error is zero. The second variant is an indirect, rank-based prioritization where $p_i = \frac{1}{\mathrm{rank}(i)}$, where $\mathrm{rank}(i)$ is the rank of transition i when the replay memory is sorted according to $|\delta_i|$. In this case,

Schaul, Tom, et al. "Prioritized experience replay." *arXiv preprint arXiv:1511.05952* (2015).

Dueling Network Architectures for Deep Reinforcement Learning

Ziyu Wang ZIYU@GOOGLE.COM
Tom Schaul SCHAUL@GOOGLE.COM
Matteo Hessel MTTHSS@GOOGLE.COM
Hado van Hasselt HADO@GOOGLE.COM
Marc Lanctot LANCTOT@GOOGLE.COM
Nando de Freitas NANDODEFREITAS@GMAIL.COM
Google DeepMind, London, UK

Abstract

In recent years there have been many successes of using deep representations in reinforcement learning. Still, many of these applications use

In spite of this, most of the approaches for RL use standard neural networks, such as convolutional networks, MLPs, LSTMs and autoencoders. The focus in these recent advances has been on designing improved control and RL al-

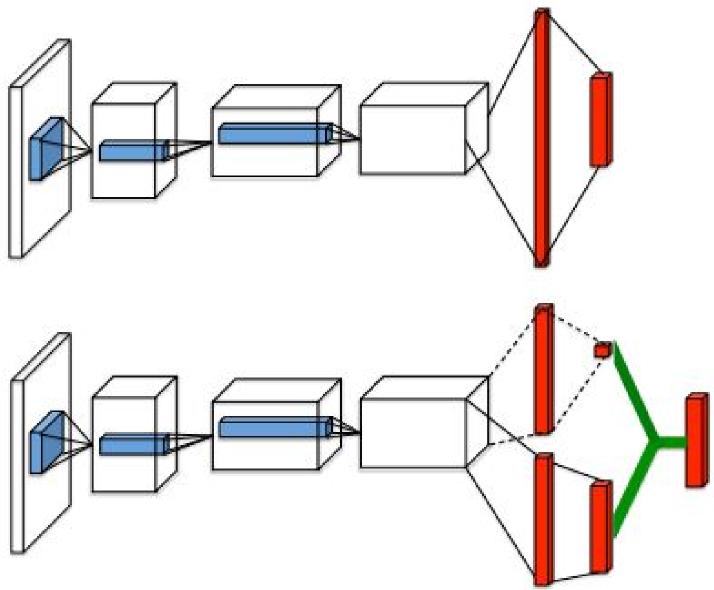


Figure 1. A popular single stream Q-network (**top**) and the dueling Q-network (**bottom**). The dueling network has two streams to separately estimate (scalar) state-value and the advantages for each action; the green output module implements equation (9) to combine them. Both networks output Q-values for each action.

$$L_i(\theta_i) = \mathbb{E}_{s,a,r,s'}\left[\left(y_i^{DQN} - Q(s,a;\theta_i)\right)^2\right],$$

$$y_i^{DQN} = r + \gamma \max_{a'} Q(s',a';\theta^-),$$

We define another important quantity, the *advantage function*, relating the value and Q functions:

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s). \tag{3}$$

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

Wang, Ziyu, Nando de Freitas, and Marc Lanctot. "Dueling network architectures for deep reinforcement learning." *arXiv preprint arXiv:1511.06581* (2015).

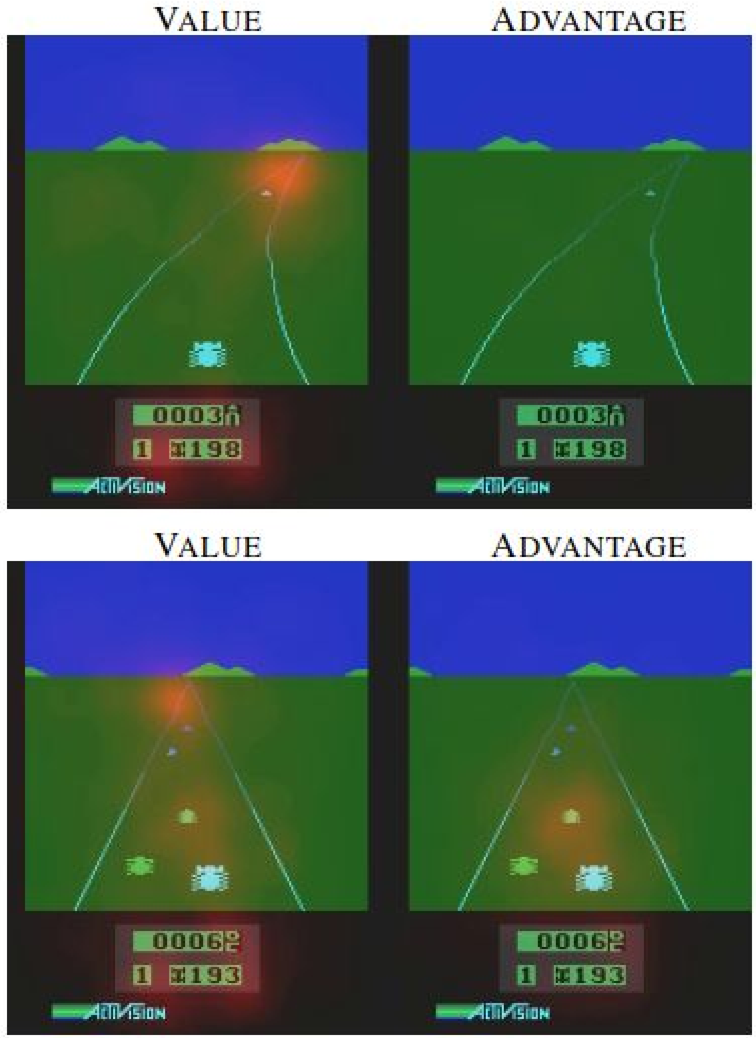


Figure 2. See, attend and drive: Value and advantage saliency maps (red-tinted overlay) on the Atari game Enduro, for a trained dueling architecture. The value stream learns to pay attention to the road. The advantage stream learns to pay attention only when there are cars immediately in front, so as to avoid collisions.

Wang, Ziyu, Nando de Freitas, and Marc Lanctot. "Dueling network architectures for deep reinforcement learning." *arXiv preprint arXiv:1511.06581* (2015).

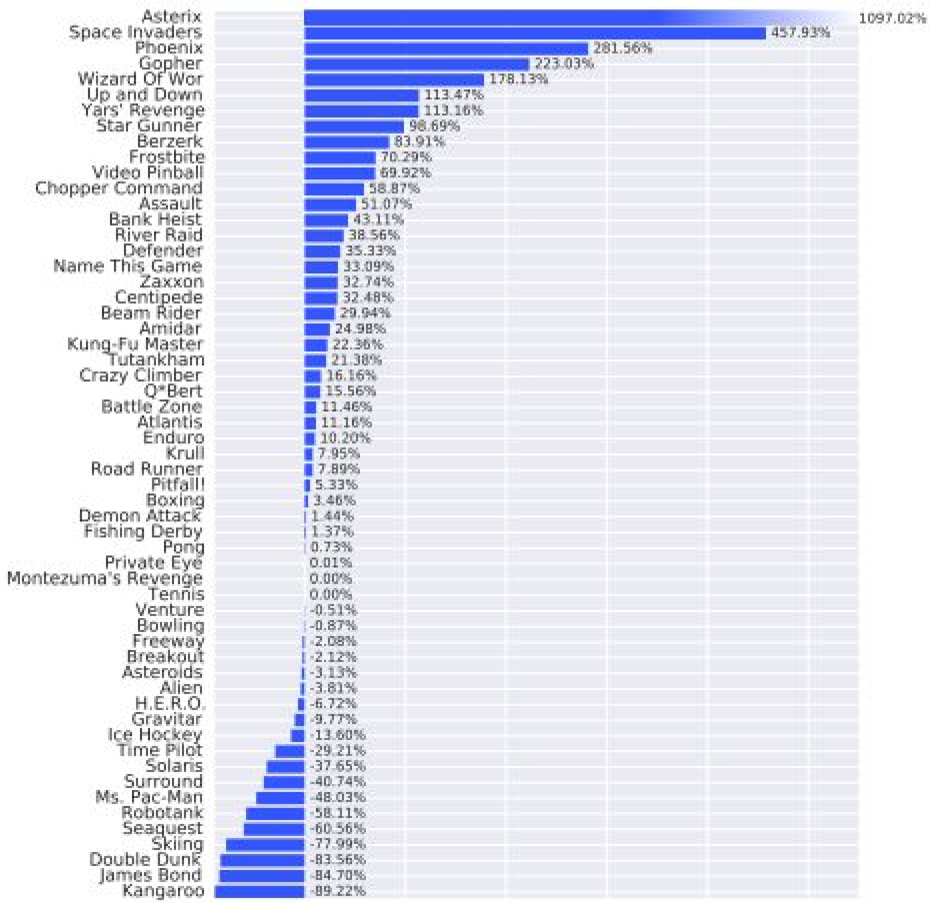
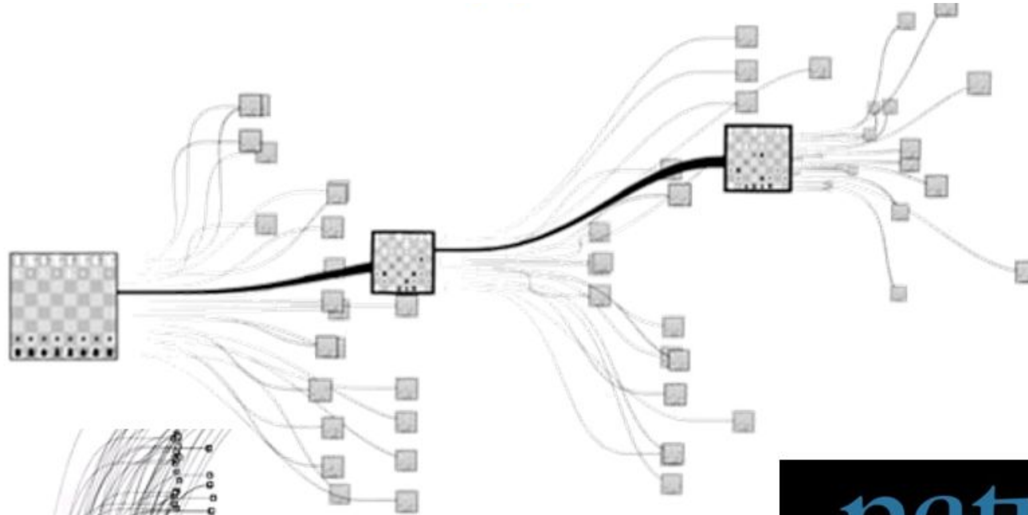
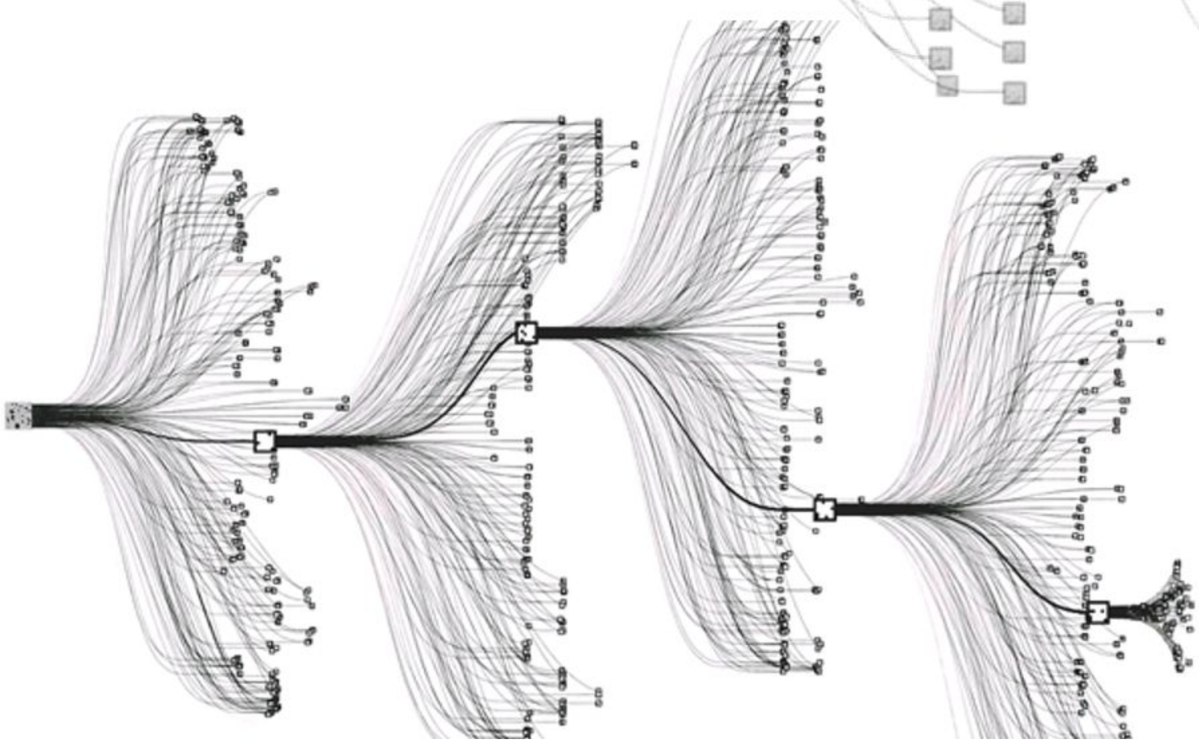


Figure 5. Improvements of dueling architecture over Prioritized DDQN baseline, using the same metric as Figure 4. Again, the dueling architecture leads to significant improvements over the single-stream baseline on the majority of games.

Wang, Ziyu, Nando de Freitas, and Marc Lanctot. "Dueling network architectures for deep reinforcement learning." *arXiv preprint arXiv:1511.06581* (2015).

ЧТО ОБЪЕДИНЯЕТ ЭТИХ ДВУХ ЛЮДЕЙ?






nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
At last — a computer program that can beat a champion Go player PAGE 484
ALL SYSTEMS GO
CONSERVATION
SONGBIRDS À LA CARTE
Illegal harvest of millions of Mediterranean birds
PAGE 452
RESEARCH ETHICS
SAFEGUARD TRANSPARENCY
Don't let openness backfire on individuals
PAGE 459
POPULAR SCIENCE
WHEN GENES GOT 'SELFISH'
Dawkins's calling card forty years on
PAGE 462
NATURE.COM/NATURE

AlphaGo

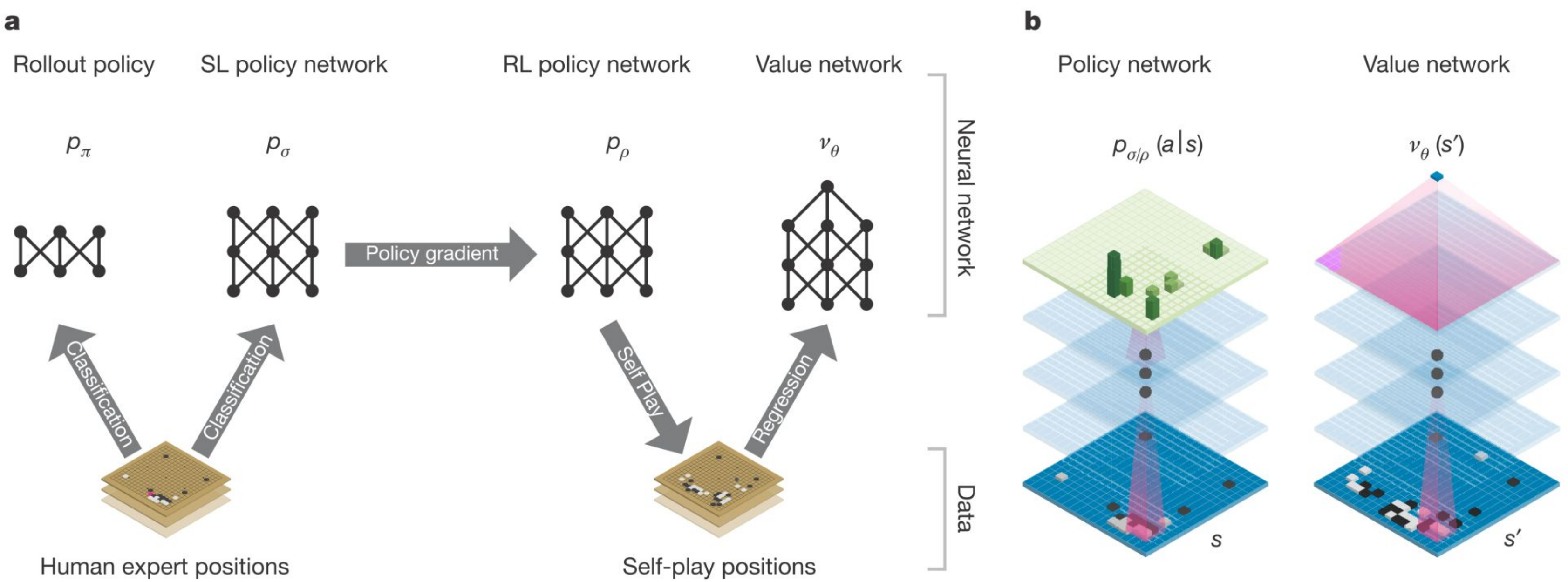


Figure 1 | Neural network training pipeline and architecture. a, A fast rollout policy p_π and supervised learning (SL) policy network p_σ are trained to predict human expert moves in a data set of positions. A reinforcement learning (RL) policy network p_ρ is initialized to the SL policy network, and is then improved by policy gradient learning to maximize the outcome (that is, winning more games) against previous versions of the policy network. A new data set is generated by playing games of self-play with the RL policy network. Finally, a value network v_θ is trained by regression to predict the expected outcome (that is, whether the current player wins) in positions from the self-play data set. **b**, Schematic representation of the neural network architecture used in AlphaGo. The policy network takes a representation of the board position s as its input, passes it through many convolutional layers with parameters σ (SL policy network) or ρ (RL policy network), and outputs a probability distribution $p_\sigma(a|s)$ or $p_\rho(a|s)$ over legal moves a, represented by a probability map over the board. The value network similarly uses many convolutional layers with parameters θ, but outputs a scalar value $v_\theta(s')$ that predicts the expected outcome in position s'.

ПОИСК ПО ДЕРЕВУ

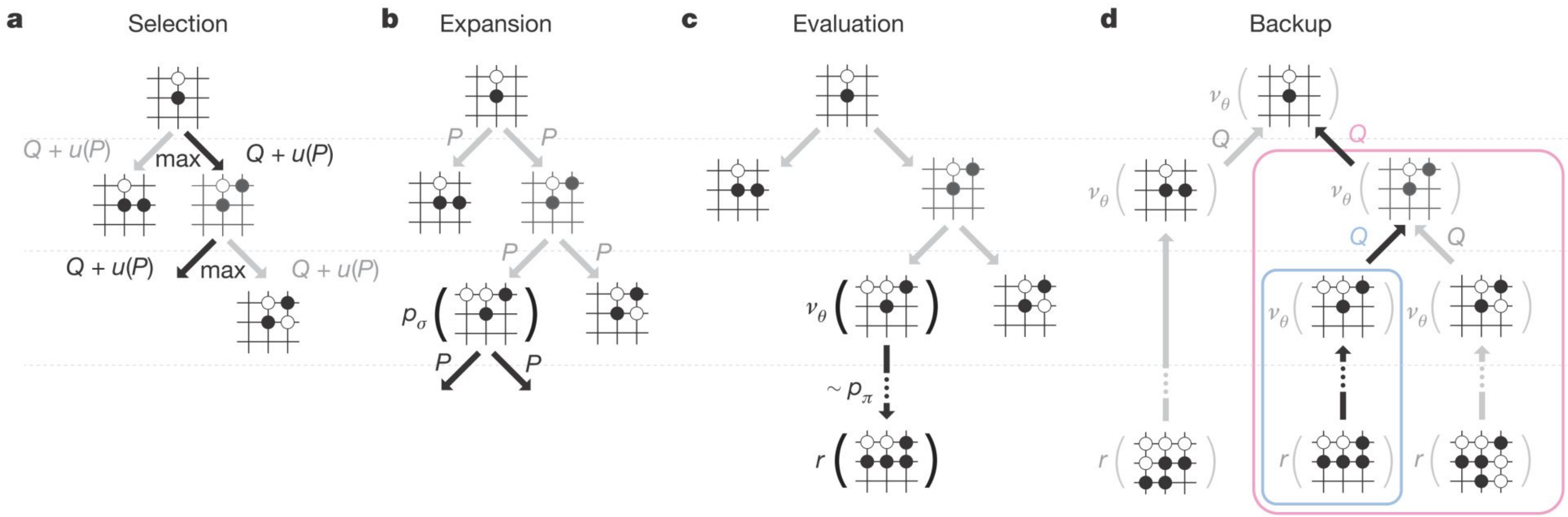


Figure 3 | Monte Carlo tree search in AlphaGo. a, Each simulation traverses the tree by selecting the edge with maximum action value Q, plus a bonus $u(P)$ that depends on a stored prior probability P for that edge. **b**, The leaf node may be expanded; the new node is processed once by the policy network p_σ and the output probabilities are stored as prior probabilities P for each action. **c**, At the end of a simulation, the leaf node is evaluated in two ways: using the value network v_θ; and by running a rollout to the end of the game with the fast rollout policy p_π, then computing the winner with function r. **d**, Action values Q are updated to track the mean value of all evaluations $r(\cdot)$ and $v_\theta(\cdot)$ in the subtree below that action.

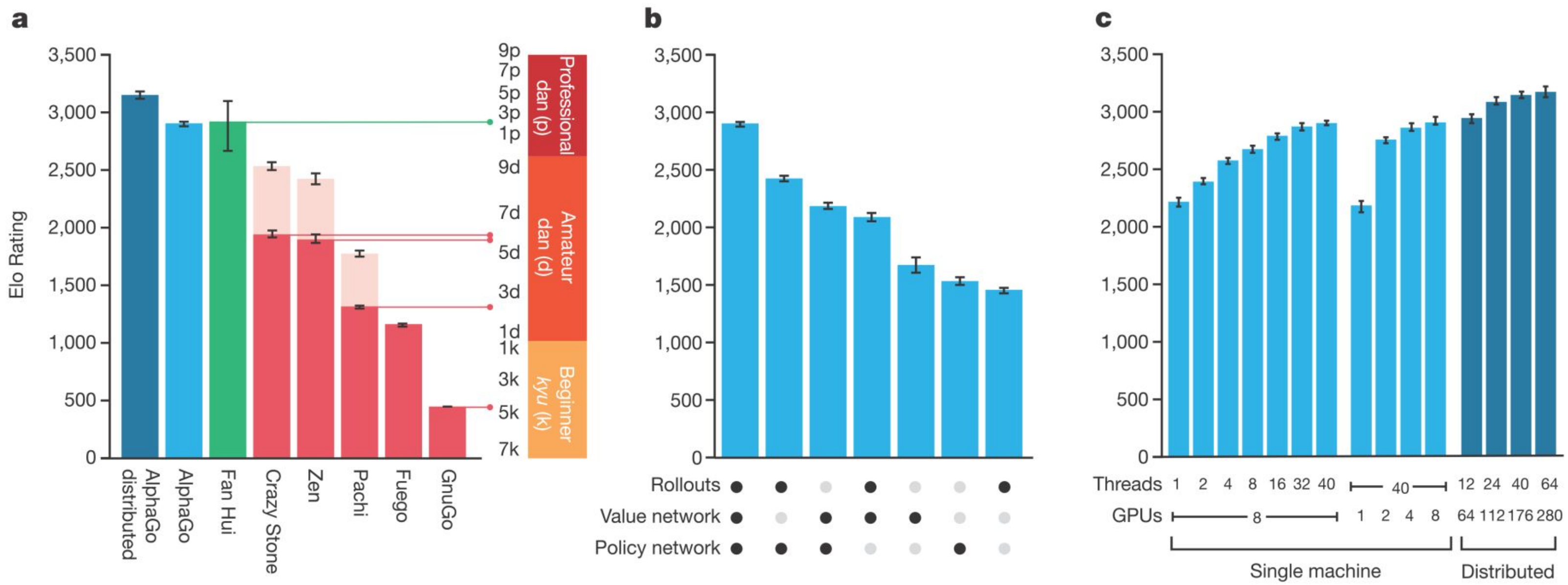


Figure 4 | Tournament evaluation of AlphaGo. **a**, Results of a tournament between different Go programs (see Extended Data Tables 6–11). Each program used approximately 5 s computation time per move. To provide a greater challenge to AlphaGo, some programs (pale upper bars) were given four handicap stones (that is, free moves at the start of every game) against all opponents. Programs were evaluated on an Elo scale[37]: a 230 point gap corresponds to a 79% probability of winning, which roughly corresponds to one amateur *dan* rank advantage on KGS[38]; an approximate correspondence to human ranks is also shown, horizontal lines show KGS ranks achieved online by that program. Games against the human European champion Fan Hui were also included; these games used longer time controls. 95% confidence intervals are shown. **b**, Performance of AlphaGo, on a single machine, for different combinations of components. The version solely using the policy network does not perform any search. **c**, Scalability study of MCTS in AlphaGo with search threads and GPUs, using asynchronous search (light blue) or distributed search (dark blue), for 2 s per move.

DeepMind AlphaGo vs Lee Sedol

4 : 1


COMPUTERWORLD FROM IDG

Sign In

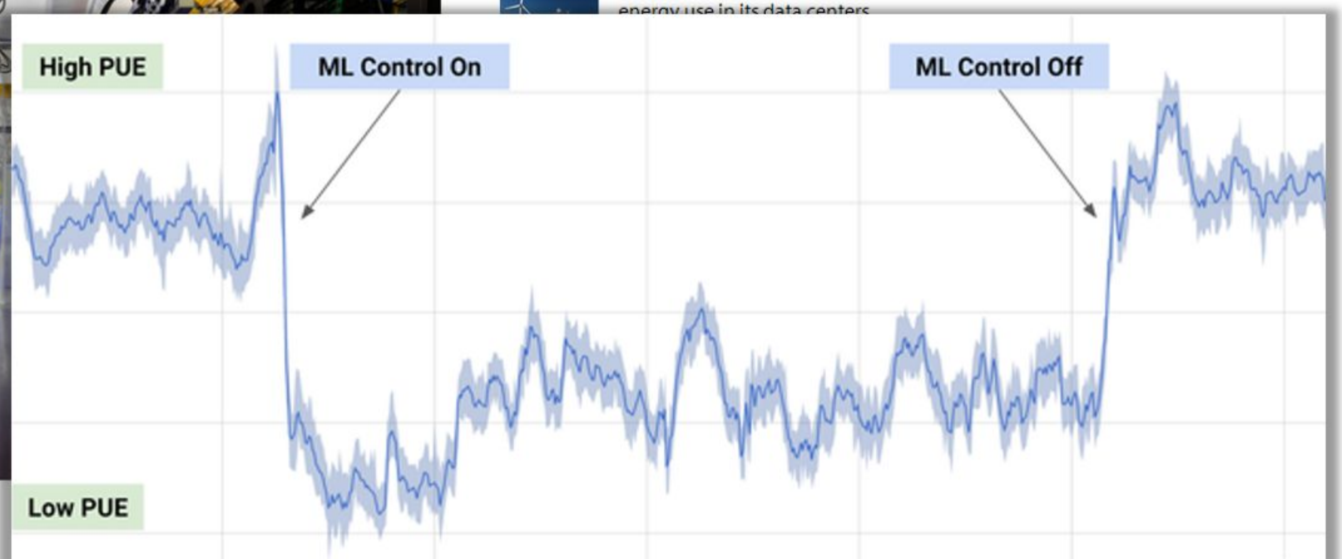
Home > Data Center

NEWS

Google's DeepMind A.I. can slash data center power use 40%

MORE LIKE THIS

Microsoft to boost renewable



Спасибо за внимание!