Deep Reinforcement Learning with Double Q-Learning

Qing Lyu

11/07/2018

Reinforcement learning by Google DeepMind

Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Deep Reinforcement Learning with Double Q-Learning

Hado van Hasselt, **Arthur Guez**, and **David Silver**
Google DeepMind

Martin Riedmiller[1], Andreas K. Fidjeland[1], Georg Ostrovski[1], Stig Petersen[1], Charles Beattie[1], Amir Sadik[1], Ioannis Antonoglou[1], Helen King[1], Dharshan Kumaran[1], Daan Wierstra[1], Shane Legg[1] & Demis Hassabis[1]

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.

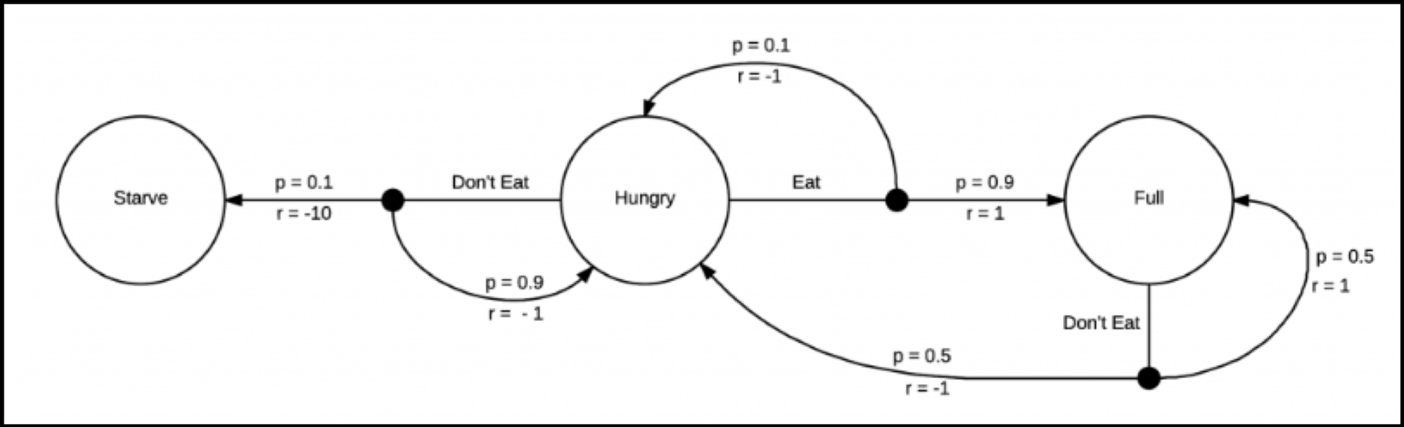
Background

- Cumulative Reward

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

- Policy
  - It is a function that takes in a state and an action and returns the probability of taking that action in that state.

$$\sum_a \pi(s,a) = 1$$



Background

- Value Functions
  - **<u>state value function</u>:** the value of a state when following a policy

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[R_t | s_t = s\right]$$

  - **<u>action value function</u>:** the value of taking an action in some state when following a certain policy

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}\left[R_t | s_t = s, a_t = a\right]$$

Background

- **Q-learning algorithm**
  - Before learning begins, *Q* is initialized to a possibly arbitrary fixed value (chosen by the programmer). Then, at each time *t* the agent selects an action a_t, observes a reward r_t, enters a new state s_{t+1} (that may depend on both the previous state s_t and the selected action), and *Q* is updated. The core of the algorithm is a simple value iteration update, using the weighted average of the old value and the new information:

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

Example: flappy bird

Two actions:
Fly upward or do nothing

Reward:
Keep alive: +1
Dead: -1000
Pass through a tube: +50

state	Fly	Do nothing
(dx_1, dy_1)	0	0
(dx_1, dy_2)	0	0
…	…	…
(dx_m, dy_{n-1})	0	0
(dx_m, dy_n)	0	0

Background

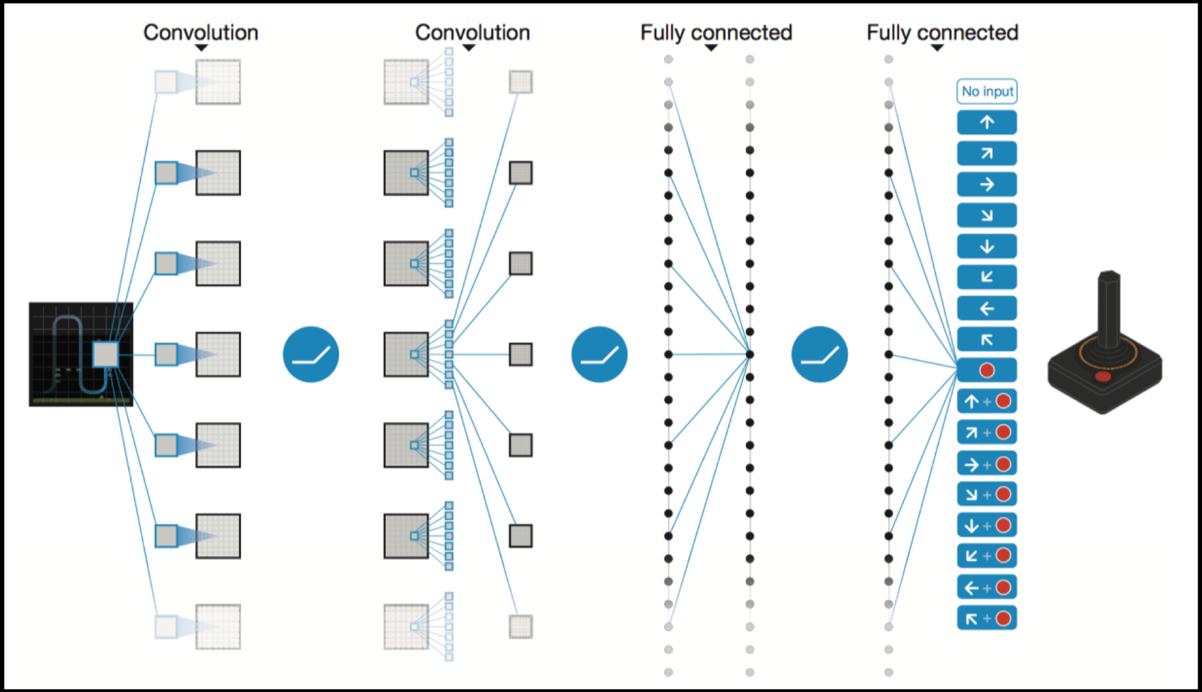
- Deep Q-learning Network
    - target network
    - Experience replay

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim \mathrm{U}(D)}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i)\right)^2\right]$$

3 convolution layers and 2 fully-connected layers

Input: 84 × 84 × 4 image

Output: a fully-connected linear layer with a single output for each valid action



Problems

- Q-learning algorithm tends to overestimate action values under certain conditions
- These overestimations may harm performance

Reason

- The max operator uses the same values both to select and to evaluate an action

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

Double Q-learning algorithm

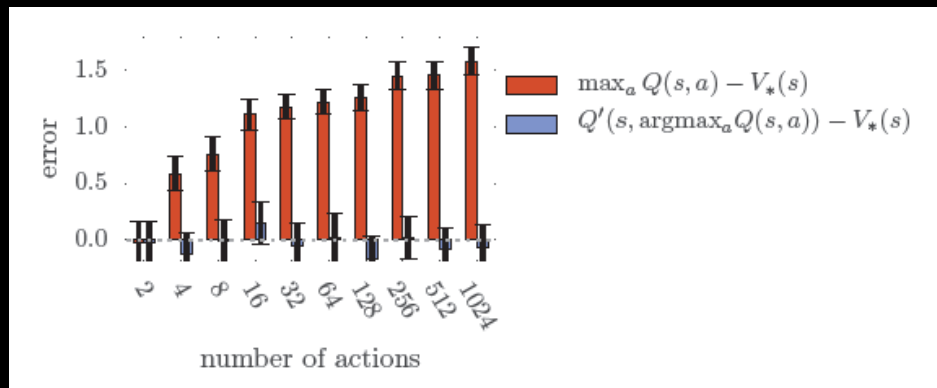
$$Y_t^{\mathrm{Q}} \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \boldsymbol{\theta}_t)$$

$$\Downarrow$$

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

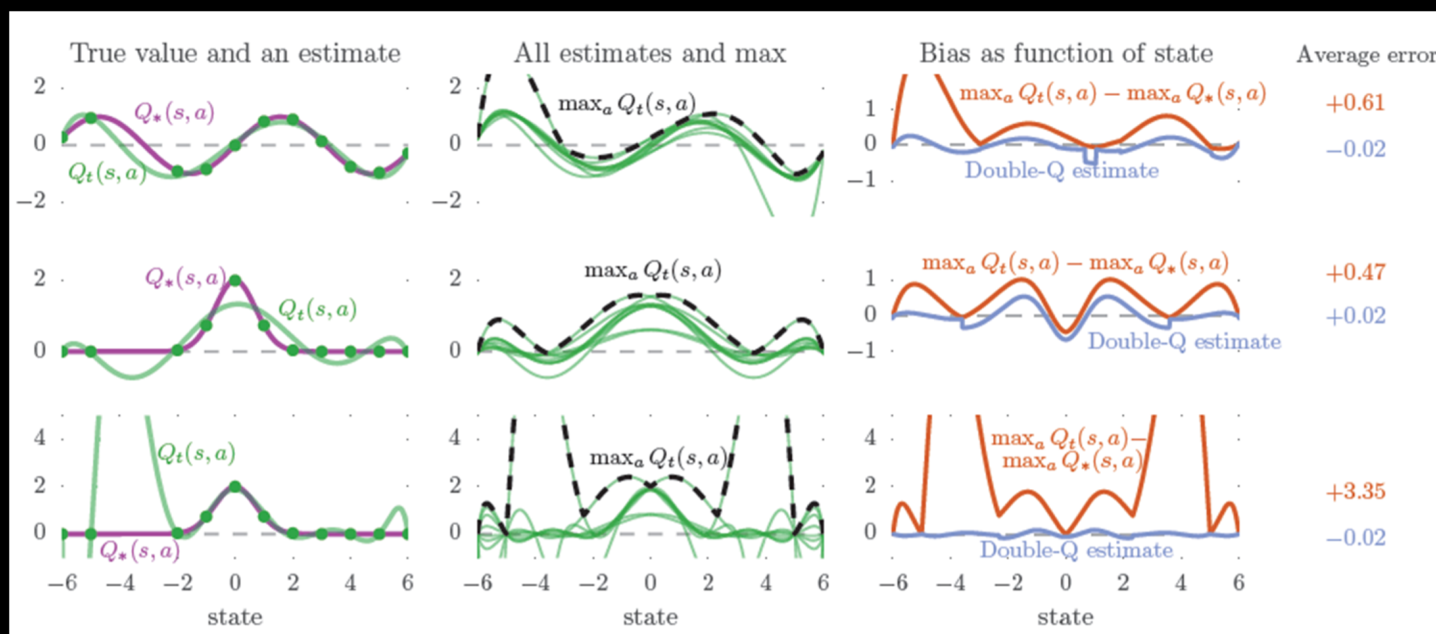
decomposing the max operation in the target into action selection and action evaluation

Double Q-learning algorithm can decrease overestimation errors



$Q(s,a) = \sin(s)$

$Q(s,a) = \ 2e^{-s^2}$



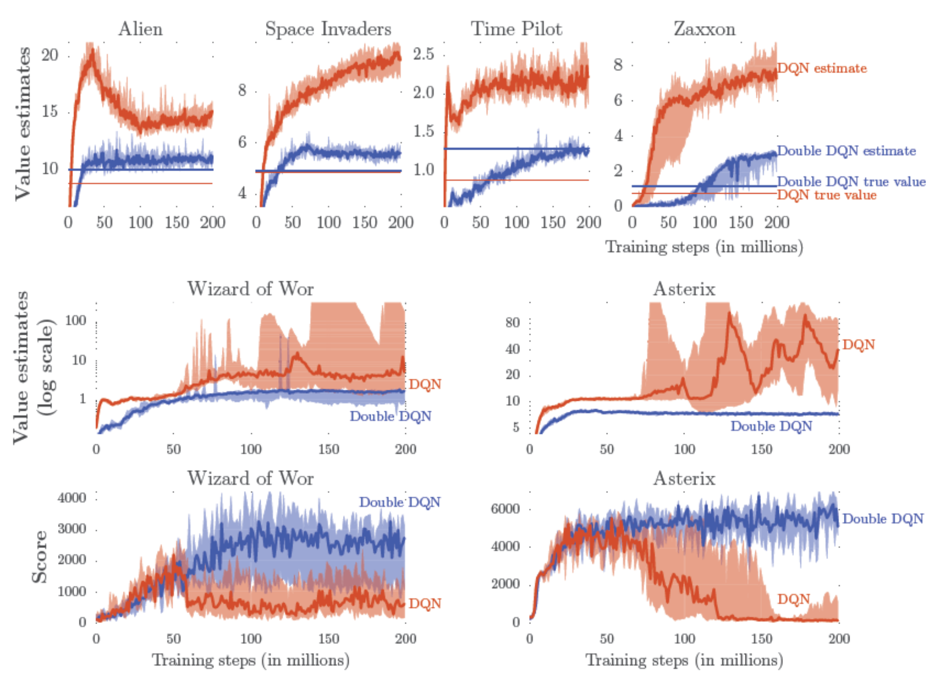
Double Deep Q-learning Network

$$C(\Theta) = \sum_i \left[R^{(i)} + \gamma \max_{\boldsymbol{a}'} f_{Q^*}(\boldsymbol{s}^{(i+1)}, \boldsymbol{a}'; \Theta^-) - f_{Q^*}(\boldsymbol{s}^{(i)}, \boldsymbol{a}^{(i)}; \Theta) \right]^2$$

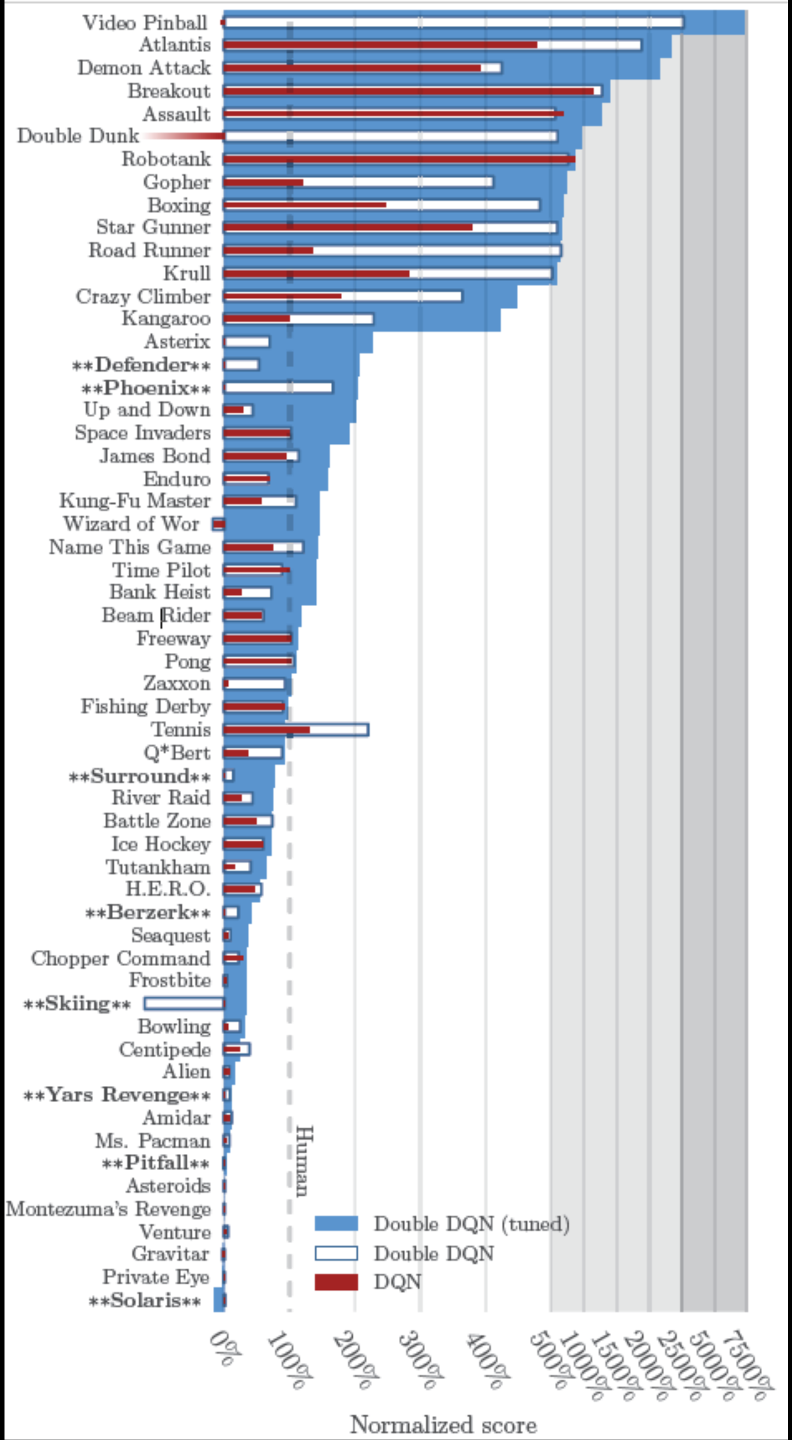
$$\Downarrow$$

$$C(\Theta) = \sum_i \left[R^{(i)} + \gamma f_{Q^*}(\boldsymbol{s}^{(i+1)}, \arg\max_{\boldsymbol{a}'} f_{Q^*}(\boldsymbol{s}^{(i+1)}, \boldsymbol{a}'; \Theta); \Theta^-) - f_{Q^*}(\boldsymbol{s}^{(i)}, \boldsymbol{a}^{(i)}; \Theta) \right]^2$$

Result



Alien
Space Invaders
Time Pilot
Zaxxon
Value estimates
DQN estimate
Double DQN estimate
Double DQN true value
DQN true value
Training steps (in millions)
Wizard of Wor
Asterix
Value estimates (log scale)
DQN
Double DQN
Score
Training steps (in millions)



Video Pinball
Atlantis
Demon Attack
Breakout
Assault
Double Dunk
Robotank
Gopher
Boxing
Star Gunner
Road Runner
Krull
Crazy Climber
Kangaroo
Asterix
Defender
Phoenix
Up and Down
Space Invaders
James Bond
Enduro
Kung-Fu Master
Wizard of Wor
Name This Game
Time Pilot
Bank Heist
Beam Rider
Freeway
Pong
Zaxxon
Fishing Derby
Tennis
Q*Bert
Surround
River Raid
Battle Zone
Ice Hockey
Tutankham
H.E.R.O.
Berzerk
Seaquest
Chopper Command
Frostbite
Skiing
Bowling
Centipede
Alien
Yars Revenge
Amidar
Ms. Pacman
Pitfall
Asteroids
Montezuma's Revenge
Venture
Gravitar
Private Eye
Solaris
Human
Double DQN (tuned)
Double DQN
DQN
0%
100%
200%
300%
400%
500%
1000%
1500%
2000%
2500%
5000%
7500%
Normalized score

Contributions

- Shows why Q-learning can be overoptimistic in large-scale problems
- Shows that overestimations are more common and severe in practice than previously acknowledged
- Double Q-learning can be used at scale to successfully reduce this overoptimism, resulting in more stable and reliable learning
- Proposes a Double DQN
- Shows that Double DQN finds better policies, obtaining new state-ofthe-art results on the Atari 2600 domain

Related work by Google DeepMind

- Prioritized DDQN https://arxiv.org/abs/1511.05952
- Dueling DDQN https://arxiv.org/abs/1511.06581
- A3C https://arxiv.org/abs/1602.01783
- Distributional DQN https://arxiv.org/abs/1707.06887
- Noisy DQN https://arxiv.org/abs/1706.10295
- Rainbow https://arxiv.org/abs/1710.02298