Human-level Control Through Deep Reinforcement Learning (Deep Q Network)

Peidong Wang

11/13/2015

Content

- Demo
- Framework
- Remarks
- Experiment
- Discussion

Content

- Demo
- Framework
- Remarks
- Experiment
- Discussion

Demo

Demo

Content

- Demo
- Framework
- Remarks
- Experiment
- Discussion

Framework

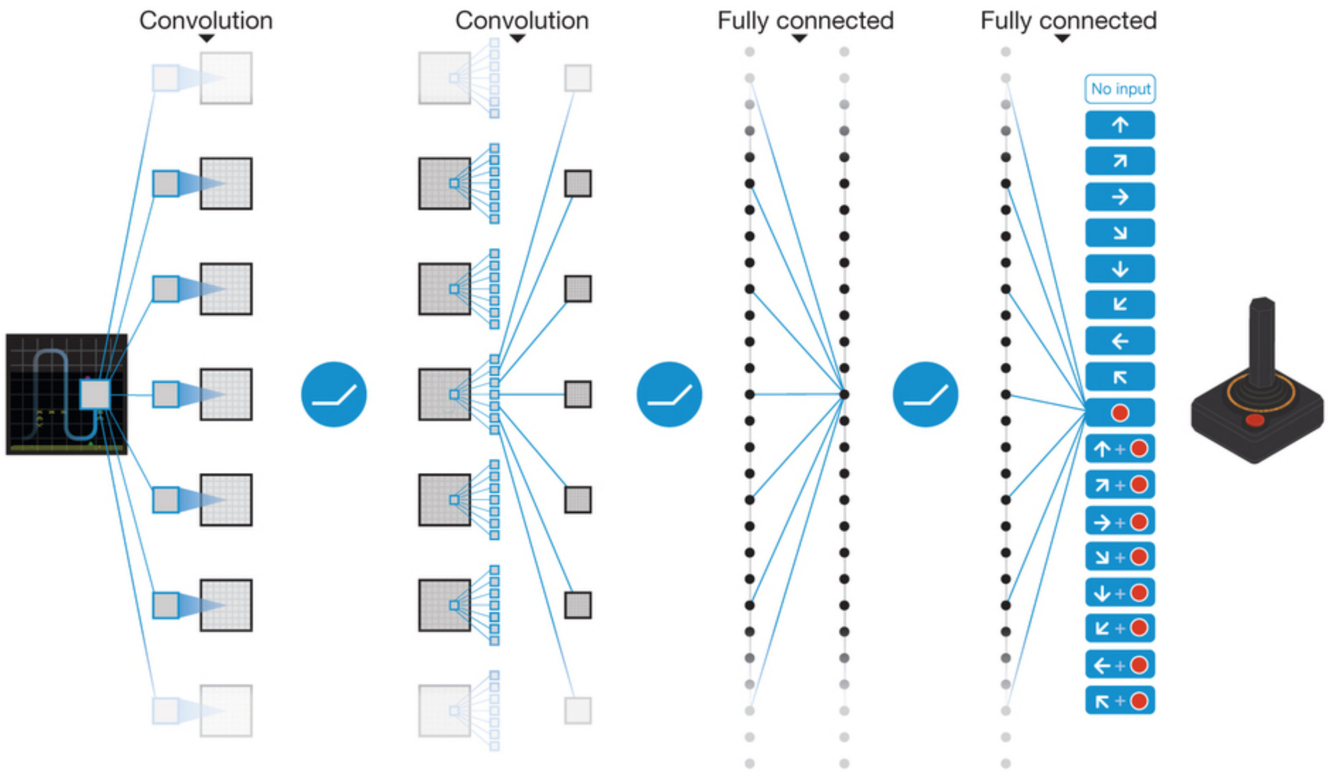


Figure from: Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al, "Human-level control through deep reinforcement learning" [J], *Nature*, Vol. 518: 529–533, February 2015.
http://www.nature.com/nature/journal/v518/n7540/full/nature14236.html

Framework

- Deep
    - Convolutional Neural Networks (CNN)
- Q (Reinforcement)
    - The network is used to approximate the Q* function. In another word, each dimension of the output corresponds to the Q value of a certain action.

Framework

- Q Function
  - What is Q?
  - It is also called action-value function in a Markov Decision Process.
  - The expected reward given observation s, action a and a certain strategy π at time t.

$$\mathrm{Q}_{\pi}(s,a)=\mathrm{E}[R_t \mid s_t = s, a_t = a]$$

  - What is Q*?
  - The maximal expected reward given observation s and action a over strategy π at time t.

$$\mathrm{Q}^{*}(s,a)=\max_{\pi}\mathrm{E}[R_t \mid s_t = s, a_t = a, \pi]$$

Framework

- Q Function
  - What is R?
  - The sum of the discounted rewards.

$$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$$

  - What is r?
  - Can be assigned manually.
  - In our application, it is the re-scaled score in a game.

Framework

- Bellman Equation
  - Note that the formula of R involves T, which is the time-step when the game terminates. So that calculating Q* directly may not work.
  - The Q* function obeys an important identity known as the Bellman Equation.

$$\mathrm{Q}^{*}(s,a)=\mathrm{E}_{s'}[r+\gamma\max_{a'}\mathrm{Q}^{*}(s',a')\,|\,s,a]$$

  - In reinforcement learning, the formula above is usually used in an iterative manner.

$$\mathrm{Q}_{i+1}(s,a)=\mathrm{E}_{s'}[r+\gamma\max_{a'}\mathrm{Q}_{i}(s',a')\,|\,s,a]$$

Framework

- Parameterization
    - Having the iteratively mannered Q* function, we can learn the optimal strategy for **each** sequence separately.
    - But we need a general model to incorporate multiple sequences. So that the Q* function needs to be parameterized by a function approximator.

$$\mathrm{Q}(s,a;\theta) \approx \mathrm{Q}^{*}(s,a)$$

- Typically, the function approximator should be linear. Deep Q Network, however, uses a deep neural network.

Framework

- Objective Function
  - Now we know that Deep Q Network uses CNN as its model and Q values as its output. And the input patterns are of course extracted from the game images. If we have the objective function and the parameter update method, we get all we need to train a model.
  - The objective function is the mean-squared error between Q* (in the Bellman Equation form) and current CNN output.

$$\mathrm{L}_i(\theta_i) = \mathrm{E}_{s,a,r}[(\mathrm{E}_{s'}[y \mid s,a] - \mathrm{Q}(s,a;\theta_i))^2] = \mathrm{E}_{s,a,r,s'}[(y - \mathrm{Q}(s,a;\theta_i))^2] + \mathrm{E}_{s,a,r}[\mathrm{V}_{s'}[y]]$$

$$y = r + \gamma \max_{a'} \mathrm{Q}(s',a';\theta_i^-)$$

Framework

- Objective Function
  - The objective function is worth seeing for a second time.

$$\mathrm{L}_i(\theta_i) = \mathrm{E}_{s,a,r}[(\mathrm{E}_{s'}[y \mid s,a] - \mathrm{Q}(s,a;\theta_i))^2] = \mathrm{E}_{s,a,r,s'}[(y - \mathrm{Q}(s,a;\theta_i))^2] + \mathrm{E}_{s,a,r}[\mathrm{V}_{s'}[y]]$$

$$y = r + \gamma \max_{a'} \mathrm{Q}(s',a';\theta_i^-)$$

  - Note that in a Deep Q Network, the targets depend on the parameters, which are network weights in this case. This is in contrast with supervised learning, where the targets are fixed before learning begins.

Framework

- Parameter Update
  - Now we get the objective function. How should we update parameters to minimize it?
  - Differentiating the objective function with respect to the parameters.

$$\nabla_{\theta_i} \mathrm{L}(\theta_i) = \mathrm{E}_{s,a,r,s'}[(r + \gamma \max_{a'} \mathrm{Q}(s',a';\theta_i^-) - \mathrm{Q}(s,a;\theta_i))\nabla_{\theta_i} \mathrm{Q}(s,a;\theta_i)]$$

  - Rather than computing the full expectation in the above gradient, it is often more computational expedient to use stochastic gradient descent.

Content

- Demo
- Framework
- Remarks
- Experiment
- Discussion

Remarks

- Many other tricks are exploited in the practical Deep Q Network.
  - **Experience replay**
  - **Periodical weights update**
  - Error term clip
  - ε-greedy action selection
  - Image preprocessing
  - Frame skipping

Remarks

- Back to the slide about parameterization, Q* function is typically parameterized using a linear approximator in reinforcement learning.
- The reason is that when using a nonlinear function approximator like a neural network, the system tends to be unstable or even to diverge.
- There are several causes.
  - The observations in a sequence are correlated.
  - Small updates to Q may significantly change the policy and therefore change the data distribution.
  - The output values and target values are correlated.

Remarks

- Experience Replay
  - For each iteration, a tuple $<Ø(s_t), a_t, r_t, Ø(s_{t+1})>$ is observed and stored in set D.
  - At the network weights update step, randomly pick up a tuple in D as the training data.

  - This method breaks the correlations in the sequence of observations.

Remarks

- Periodical Weights Updtate
  - Use a separate network $\hat{Q}$ to generate the targets.
  - After every C updates we clone the network Q to the network $\hat{Q}$ and use $\hat{Q}$ to generate targets for the following C updates to Q.

  - This method adds a delay between the time an update to Q is made and the time the update affects the targets.

Content

- Demo
- Framework
- Remarks
- Experiment
- Discussion

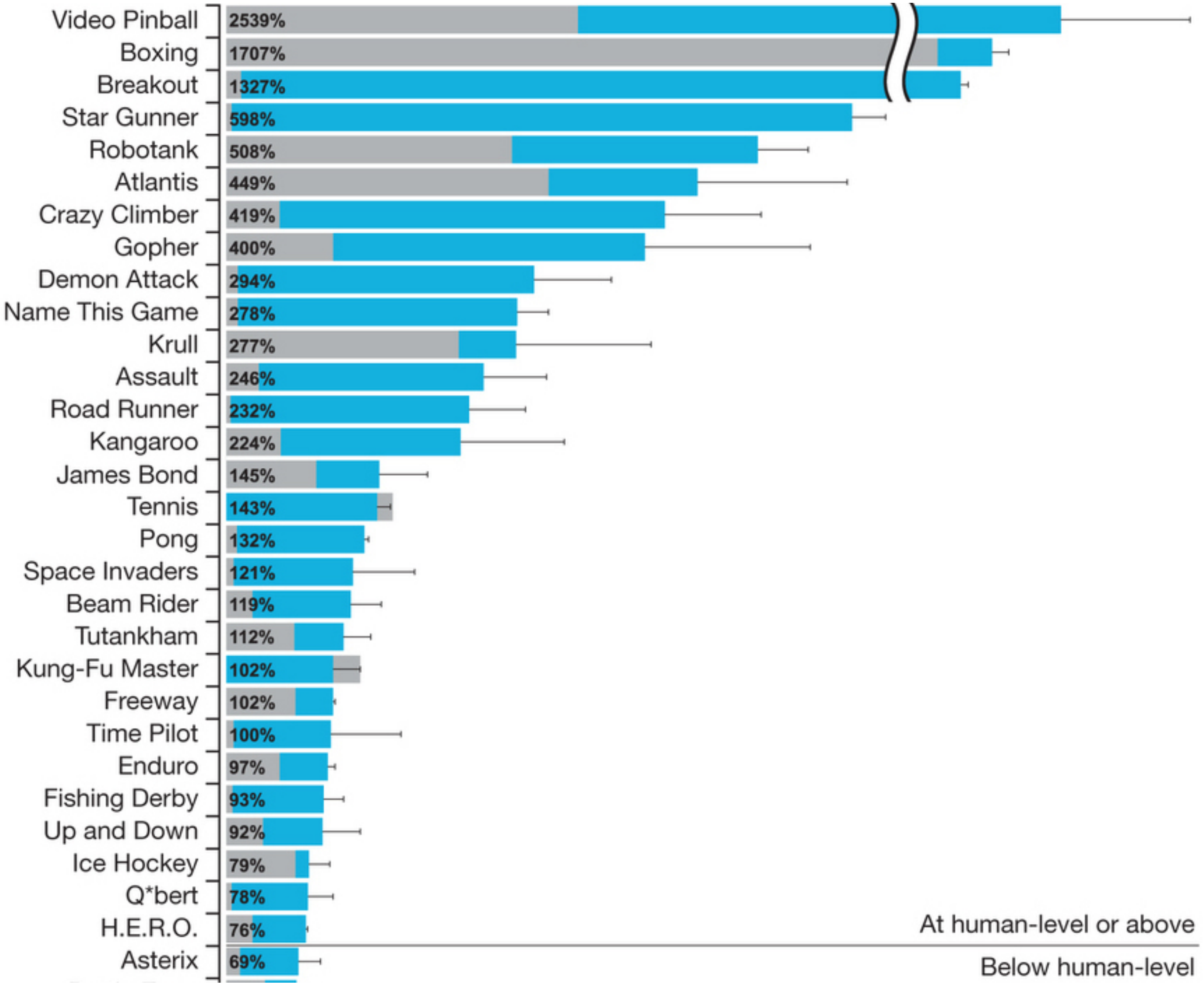
Experiment

- Experiment Setting
  - Data: Atari 2600 games (210 * 160 color video at 60Hz)
  - Training: A total of 50 million frames (around 38 days of game experience).
  - Testing: Play each game 30 times for up to 5 min each time with different initial random conditions and an ε-greedy policy with $\varepsilon = 0.05$.

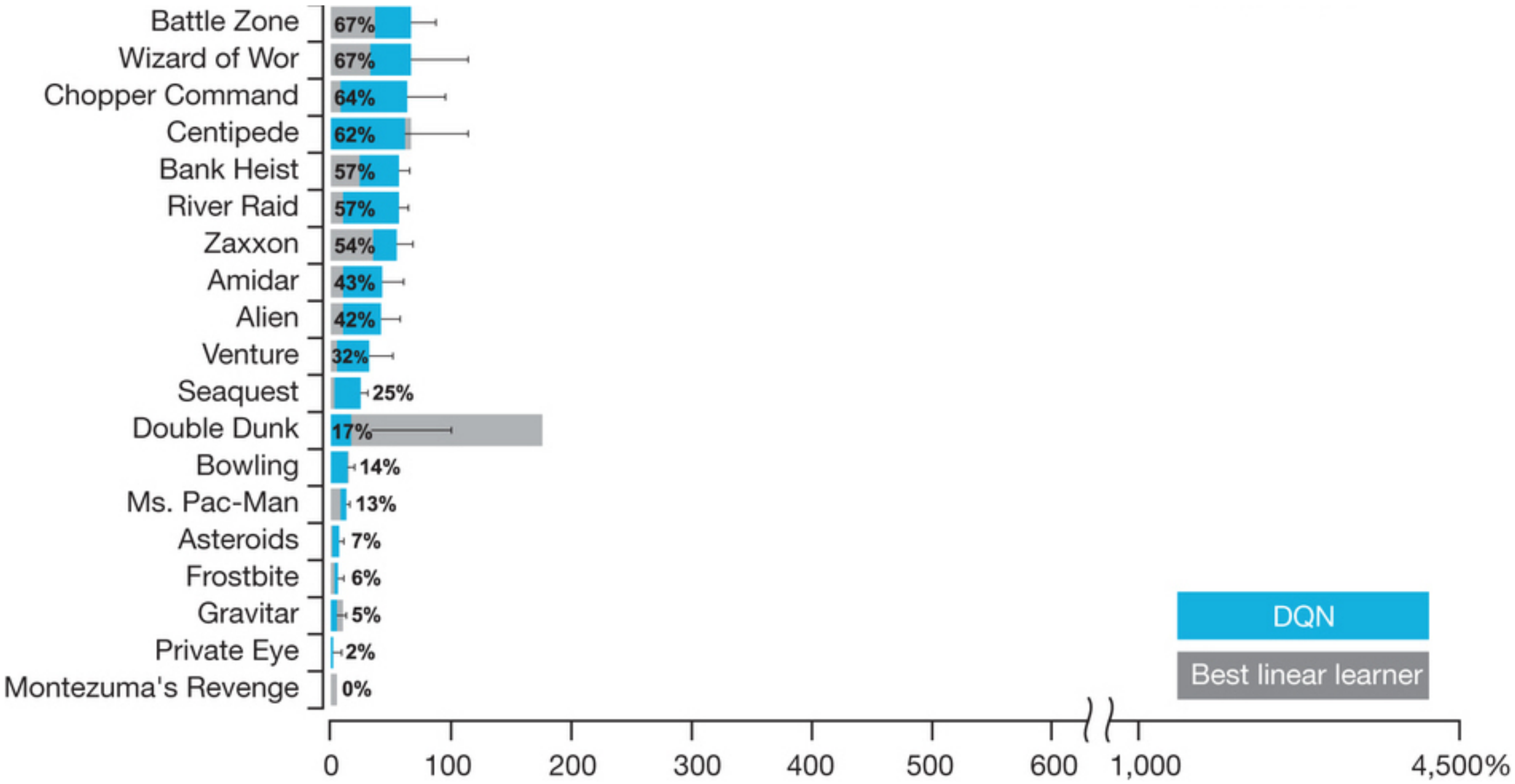
Experiment

- Result Presentation:
  - Compare the performance with human players.
  - The human performance is the average reward achieved from around 20 episodes of each game lasting a maximum of 5 min each, following around 2 h of practice playing each game.
  - A random agent is also proposed.
  - The random agent chooses an action at 10Hz, which is every sixth frame, repeating its last action on intervening frames.
  - 10Hz is about the fastest that a human player can select the ‘fire’ button.

Experiment


Video Pinball
2539%
Boxing
1707%
Breakout
1327%
Star Gunner
598%
Robotank
508%
Atlantis
449%
Crazy Climber
419%
Gopher
400%
Demon Attack
294%
Name This Game
278%
Krull
277%
Assault
246%
Road Runner
232%
Kangaroo
224%
James Bond
145%
Tennis
143%
Pong
132%
Space Invaders
121%
Beam Rider
119%
Tutankham
112%
Kung-Fu Master
102%
Freeway
102%
Time Pilot
100%
Enduro
97%
Fishing Derby
93%
Up and Down
92%
Ice Hockey
79%
Q*bert
78%
H.E.R.O.
76%
Asterix
69%
At human-level or above
Below human-level

Experiment



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
67%
67%
64%
62%
57%
57%
54%
43%
42%
32%
25%
17%
14%
13%
7%
6%
5%
2%
0%
0
100
200
300
400
500
600
1,000
4,500%
DQN
Best linear learner

Experiment

- The above two figures are from Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al, “Human-level control through deep reinforcement learning” [J], *Nature*, Vol. 518: 529–533, February 2015. http://www.nature.com/nature/journal/v518/n7540/full/nature14236.html

- The values shown in the figure are the normalized performances of Deep Q Network, which is calculated as 100 × (DQN score − random play score)/(human score − random play score).

Content

- Demo
- Framework
- Remarks
- Experiment
- Discussion

Discussion

- Deep Q Network combines reinforcement learning with deep neural networks. So that the model doesn’t need fixed labels and is expressive enough to approximate nonlinear Q* function.
- Deep Q Network shows that by viewing neural networks as tools for function approximating and embedding them into other learning methods, we may get better learning machines.
- The ideas in Deep Q Network may be used in robust speech recognition, where we can continuously modify our model through interaction with the environment.

Appendix 1: Source Program

- The source program of Deep Q Network is provided on the website.
- Hardware settings
  - A Linux server whose operating system uses apt-get command (I used a server which has been installed Ubuntu).
  - The program can be run both on GPUs and CPUs.
- Software settings
  - Need to install tools like LuaJIT, Torch 7.0, nngraph, Xitari and AleWrap.
  - Need to download Atari game files.
- Scripts
  - Changes of scripts are mainly in run_gpu/run_cpu in the root folder and train_agent.lua in the DQN folder.

Appendix 2: Reinforcement Learning

- For a quick start on Reinforcement Learning, you can use materials by David Silver on http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html.
- For a detailed study, you may read “Reinforcement Learning: An Introduction” by Richard S. Sutton and Andrew G. Barto. The online version is on https://webdocs.cs.ualberta.ca/~sutton/book/ebook/the-book.html.

Thank You!