Horizon Robotics

An Introduction to Reinforcement Learning

Chang Huang

References

- UCL Course on RL
- Richard S. Sutton and Andrew G. Barto's Book
- David Silver's slides on ICLR2015
- Andrej Karpathy blog
- Nervana's webpage
- Wikipedia

What is RL

Horizon Robotics

- Reinforcement learning is an area of machine learning inspired by behaviorist psychology, concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. https://en.wikipedia.org/wiki/Reinforcement_learning

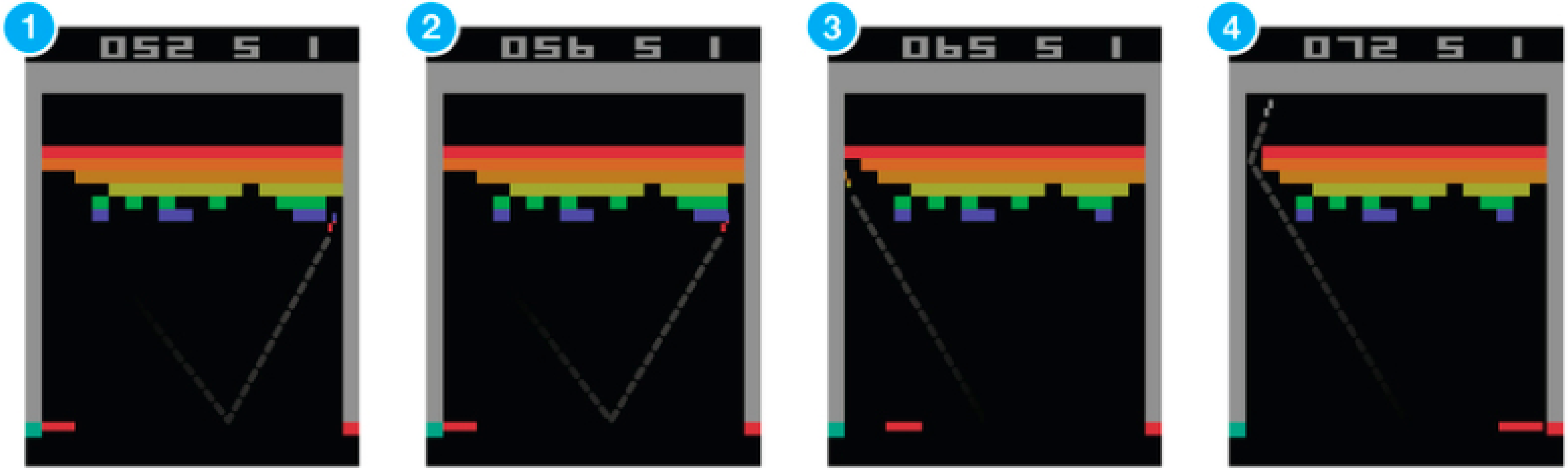
- Reinforcement learning is learning what to do--how to map situations to actions--so as to maximize a numerical reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them. http://webdocs.cs.ualberta.ca/~sutton/book/ebook/the-book.html

RL = AI ?

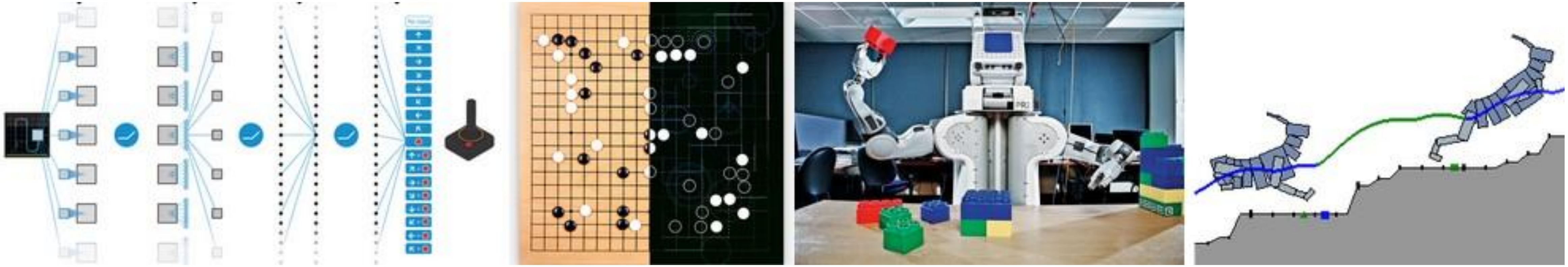
- RL is a general-purpose framework for artificial intelligence
  - RL is for an agent with the capacity to act
  - Each action influences the agent's future state
  - Success is measured by a scalar reward signal
- RL in a nutshell:
  - Select actions to maximize future reward
- We seek a single agent which can solve any human-level task
  - The essence of an intelligent agent

Breakout as an Example

Horizon Robotics



Examples of RL



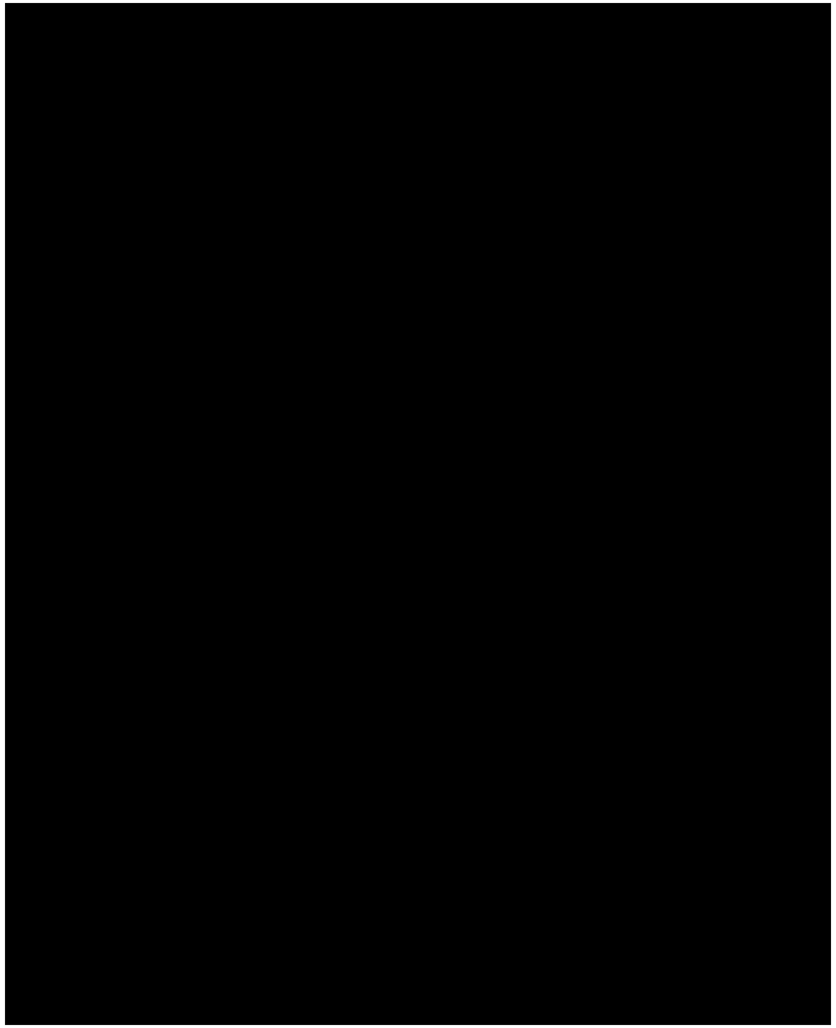
From left to right: Deep Q Learning network playing ATARI, AlphaGo, Berkeley robot stacking Legos, physically-simulated quadruped leaping over terrain.

Examples of RL

Horizon Robotics

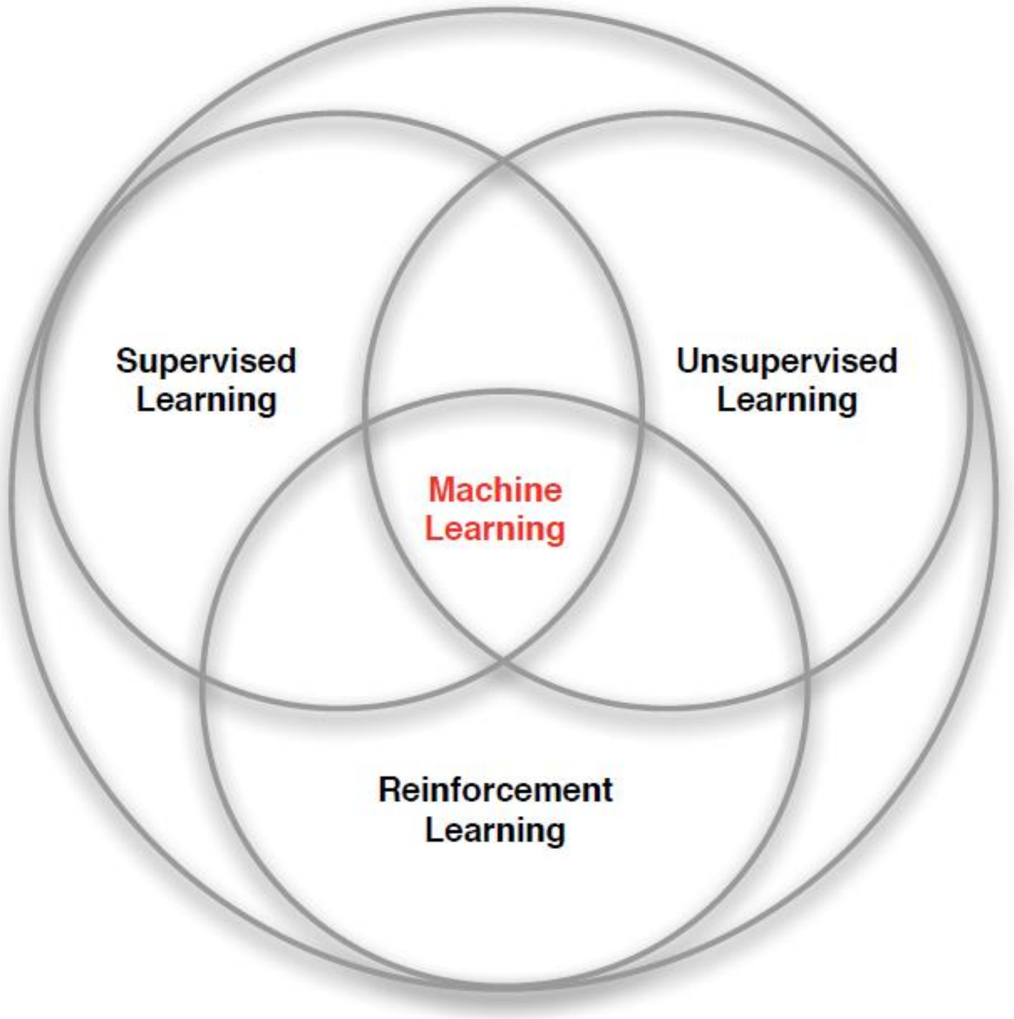
- Control physical systems: walk, fly, drive, swim, ...
- Interact with users: retain customers, personalize channel, optimize user experience, ...
- Solve logistical problems: scheduling, bandwidth allocation, elevator control, cognitive radio, power optimization, ..
- Play games: chess, checkers, Go, Atari games, ...
- Learn sequential algorithms: attention, memory, conditional computation, activations, ...

RL Plays Atari



RL Plays Doom

Branches of Machine Learning



Characteristics of RL

Horizon Robotics

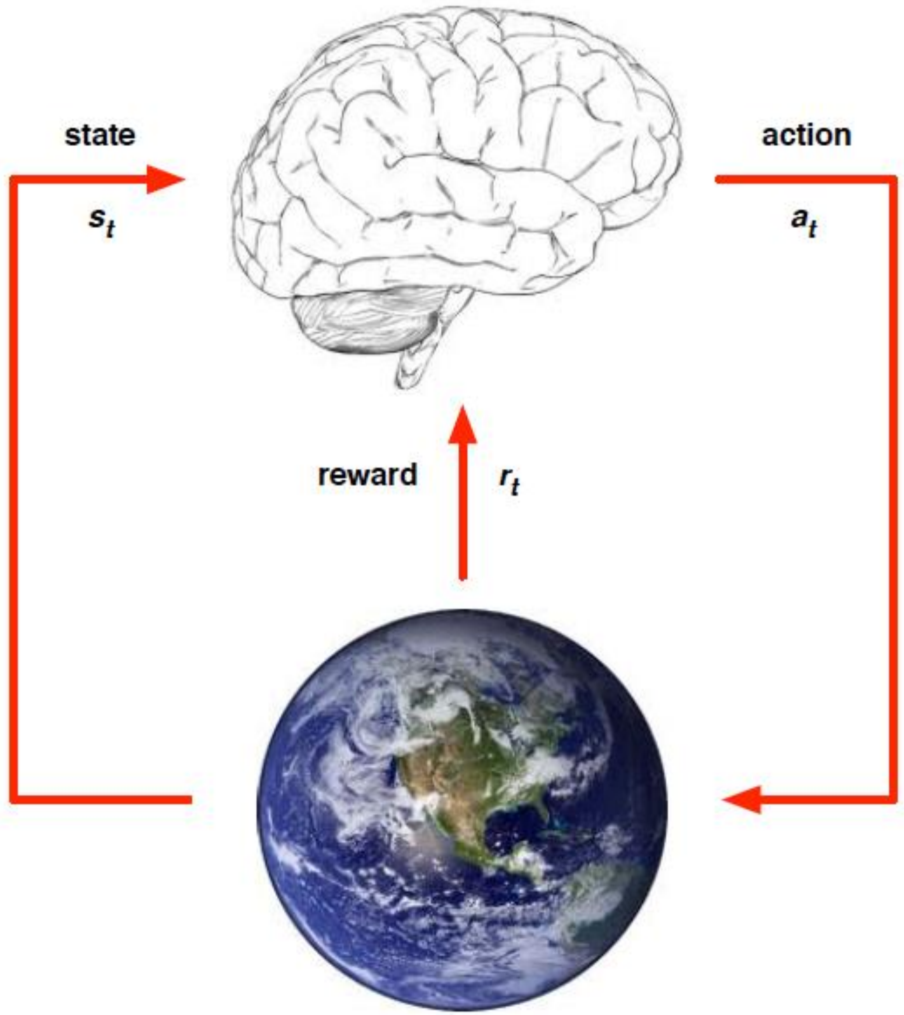
- What makes reinforcement learning different from other machine learning paradigms?
  - There is no supervisor, only a reward signal
  - Feedback is delayed, not instantaneous
  - Time really matters (sequential, non i.i.d data)
  - Agent's actions affect the subsequent data it receives
  - An agent must be able to learn from its own experience

Agent and Environment

- At each step t, the agent
  - receive state s_t
  - receive scalar reward r_t
  - execute action a_t

- The environment
  - receive action a_t
  - emit state s_{t+1}
  - emit scalar reward r_{t+1}

- Transition: $\langle s_t, a_t, r_{t+1}, s_{t+1} \rangle$



Markov Decision Process

- One episode of Markov decision process is a finite sequence of states, actions and rewards.

$$s_0, a_0, r_1, s_1, a_1, r_2, s_2, \cdots s_{n-1}, a_{n-1}, r_n, s_n$$

- Markov assumption:

$$P(s_{t+1} | s_t, a_t, s_{t-1}, a_{t-1}, \cdots, s_0, a_0) = P(s_{t+1} | s_t, a_t)$$

Discounted Future Reward

- Total reward:

$$R = r_1 + r_2 + r_3 + \cdots + r_n$$

- Total future reward:

$$R_t = r_t + r_{t+1} + r_{t+2} + \cdots + r_n$$

- Discounted future reward:

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{n-t} r_n, \quad \gamma \in [0,1]$$

Policy, Value and Transition Model

Horizon Robotics

- Policy is a behavior function choosing actions given states

$$a = \pi(s) \quad \text{or} \quad p^{\pi}(a|s)$$

- Value function is expected discounted future award, starting from a given state and performing a given action

$$Q(s_t, a_t) = \mathbb{E}(R_{t+1}|s_t, a_t)$$

- Transition model estimates the future reward and state

$$p(s_{t+1}, r_{t+1}|s_t, a_t)$$

Approaches to RL

- Value-based RL
  - Estimate the optimal value function $Q^*(s, a)$
  - This is the maximum value achievable under any policy
- Policy-based RL
  - Search directly for the optimal policy π^*
  - This is the policy achieving maximum future reward
- Model-based RL
  - Build a transition model of the environment
  - Plan (e.g. by lookahead) using model $p(s_{t+1}, r_{t+1} | s_t, a_t)$

Bellman Equation

- Optimal value function can be unrolled recursively

$$Q^*(s,a) = \mathbb{E}_{s'}\left(r + \gamma \max_{a'} Q^*(s',a') \,|s,a\right)$$

- Q-function can be iteratively updated by using the Bellman equation

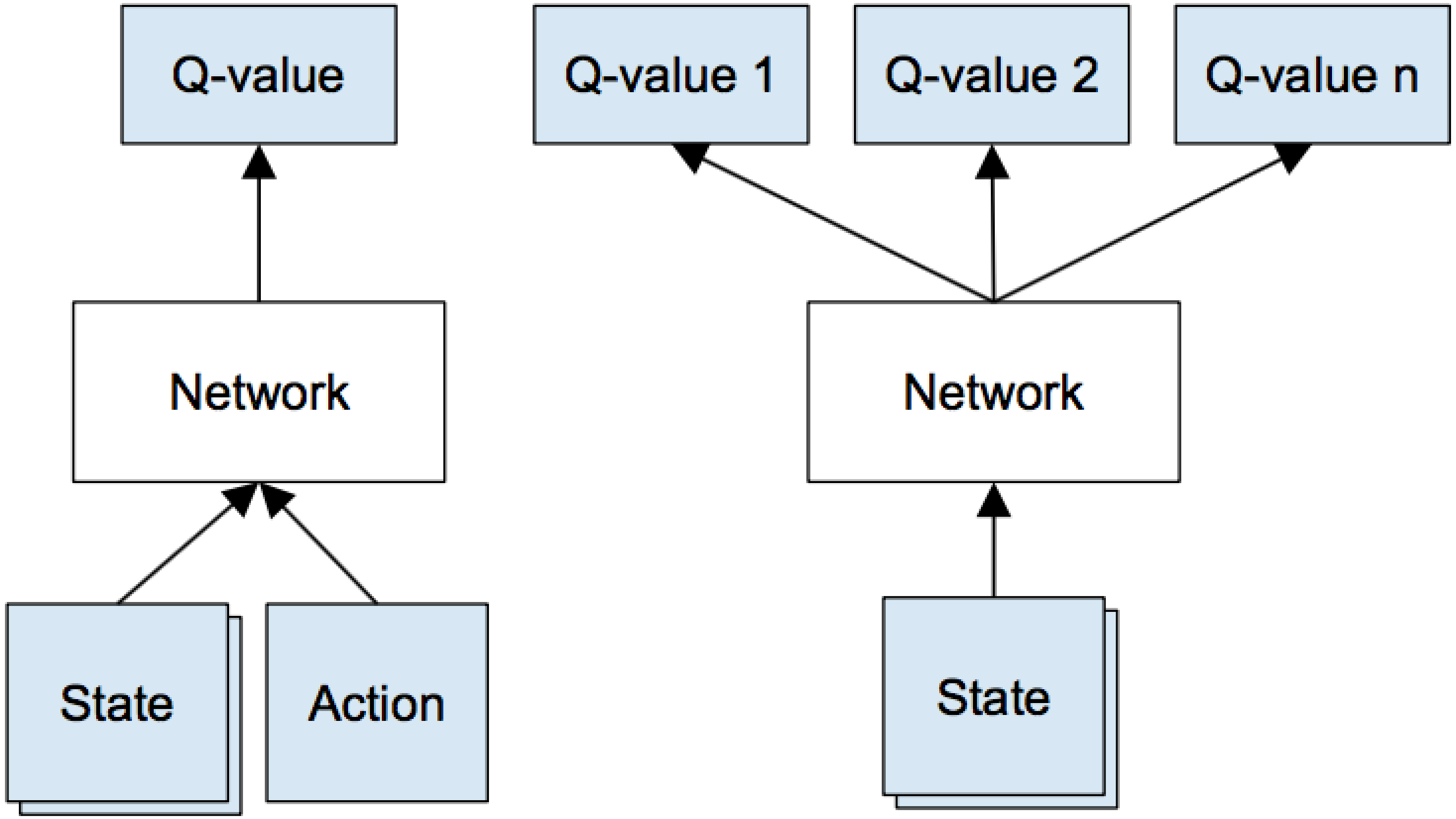
$$Q_{i+1}(s,a) \leftarrow \mathbb{E}_{s'}\left(r + \gamma \max_{a'} Q_i(s',a') \,|s,a\right)$$

Q-Learning

Horizon Robotics

```
initialize Q[num_states,num_actions] arbitrarily
observe initial state s
repeat
      select and carry out an action a
      observe reward r and new state s'
      Q[s,a] = Q[s,a] + α(r + γ max_a' Q[s',a'] - Q[s,a])
      s = s'
until terminated
```

Deep Q-Network



Deep Q-Learning

Horizon Robotics

- Represent value function by deep Q-network $Q(s, a: \theta)$
- Define a MSE loss function for Q-value approximation

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

- Optimize by SGD

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta} = \mathbb{E}_{s,a,r,s'}\left[\left(r + \gamma \max_{a'} Q(s', a': \theta) - Q(s, a: \theta)\right)\frac{\partial Q(s, a: \theta)}{\partial \theta}\right]$$

Challenges in Deep Q-Learning

Horizon Robotics

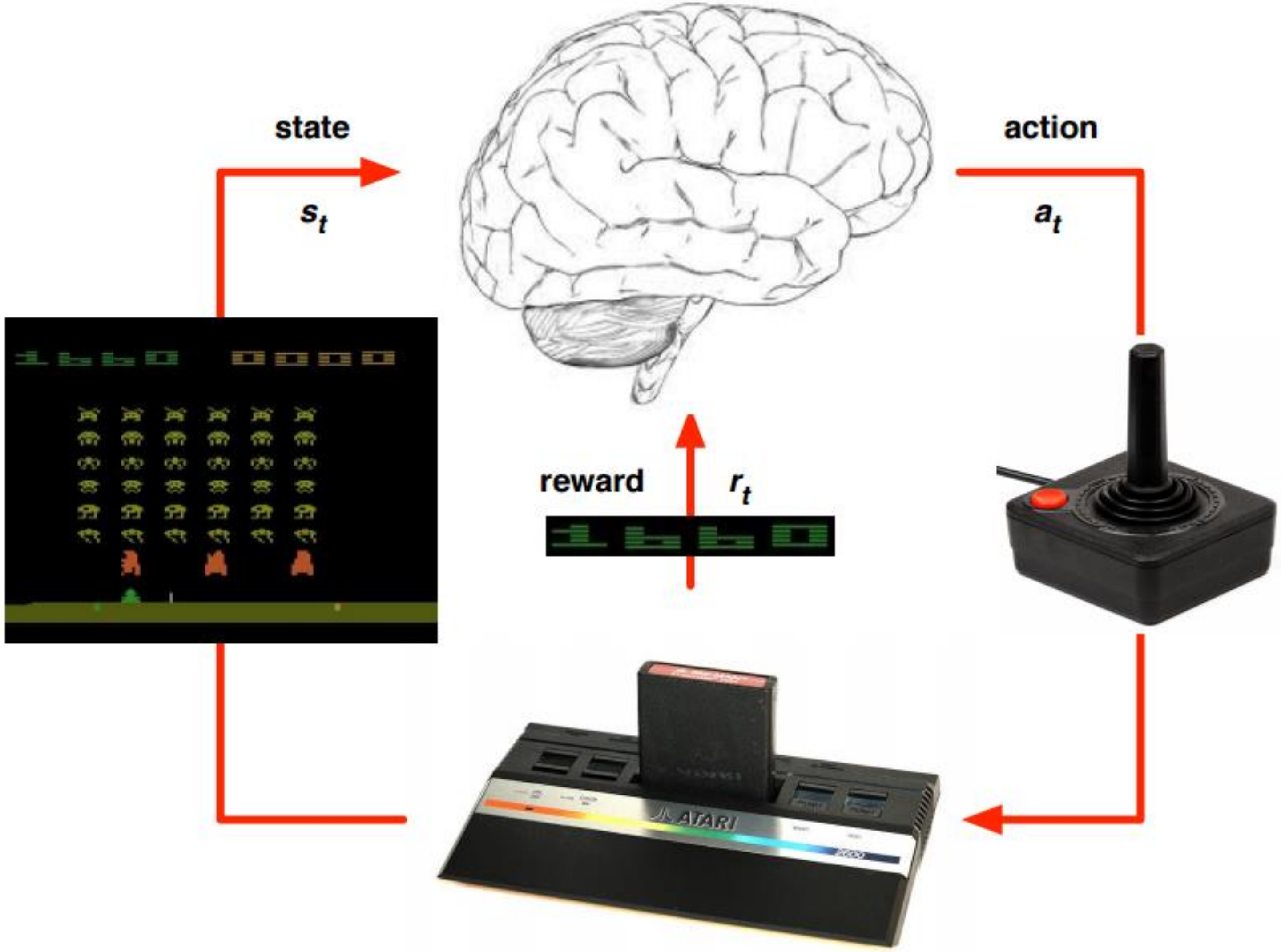
- Exploration-exploitation dilemma
  - Random exploration gradually becomes greedy and crude with exploitation of converging Q functions
- Sequential data are NOT i.i.d
  - Choosing an certain action affects the coming transitions
  - Highly correlated data are harmful for SGD method
- Non-stationary target
  - Target changes while θ is updated, causing oscillation during training

- Successful tricks:
  - ε-greedy exploration & experience replay & target Q-network

Deep Q-Learning with Experience Replay

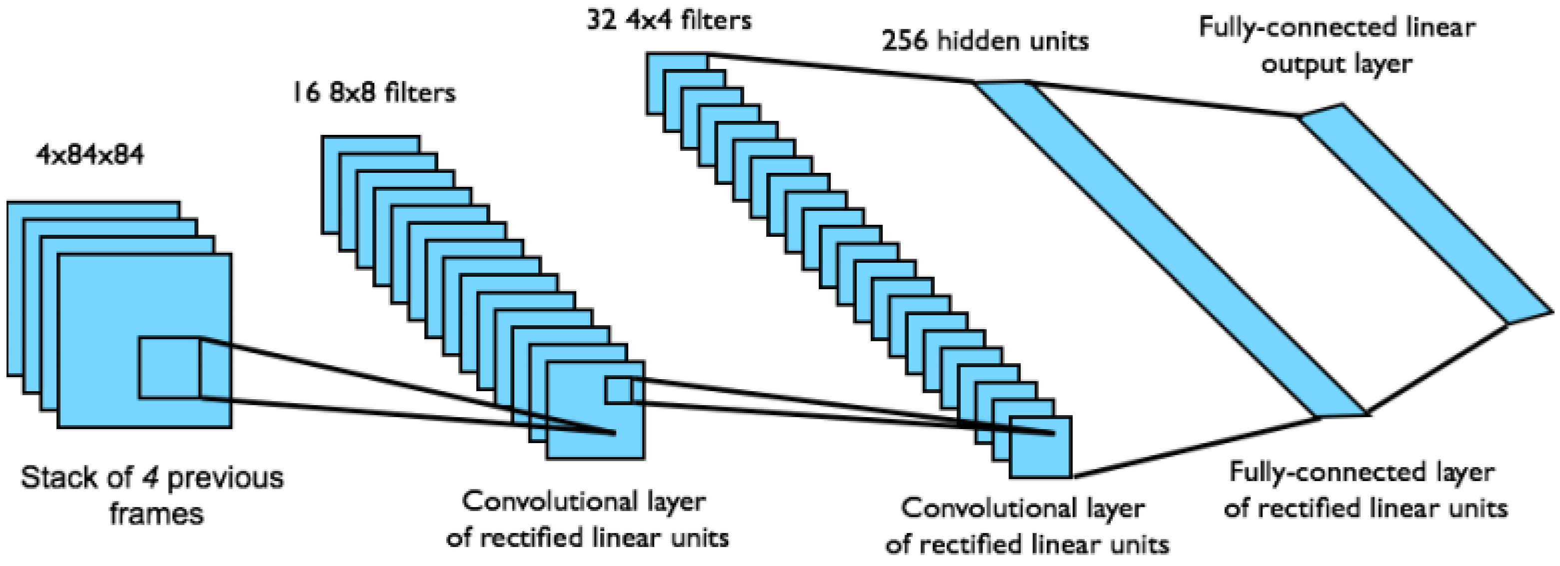
```
initialize replay memory D
initialize action-value function Q with random weights
observe initial state s
repeat
      select an action a
            with probability ε select a random action
            otherwise select a = argmax_a' Q(s,a')
      carry out action a
      observe reward r and new state s'
      store experience <s, a, r, s'> in replay memory D

      sample random transitions <ss, aa, rr, ss'> from replay memory D
      calculate target for each minibatch transition
            if ss' is terminal state then tt = rr
            otherwise tt = rr + γmax_a' Q(ss', aa')
      train the Q network using (tt - Q(ss, aa))^2 as loss

      s = s'
until terminated
```

RL in Atari

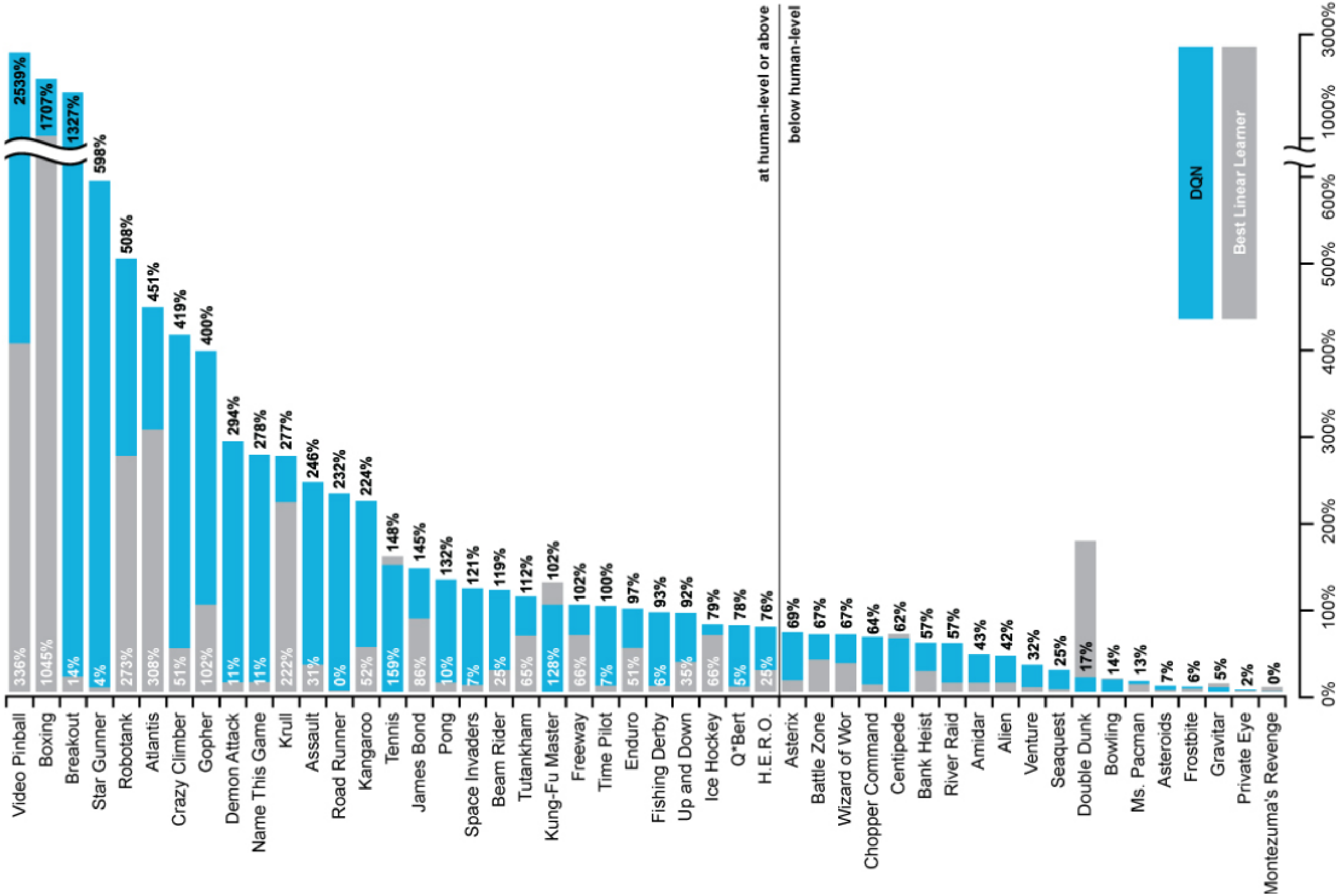


Deep Q-Learning in Atari

- 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

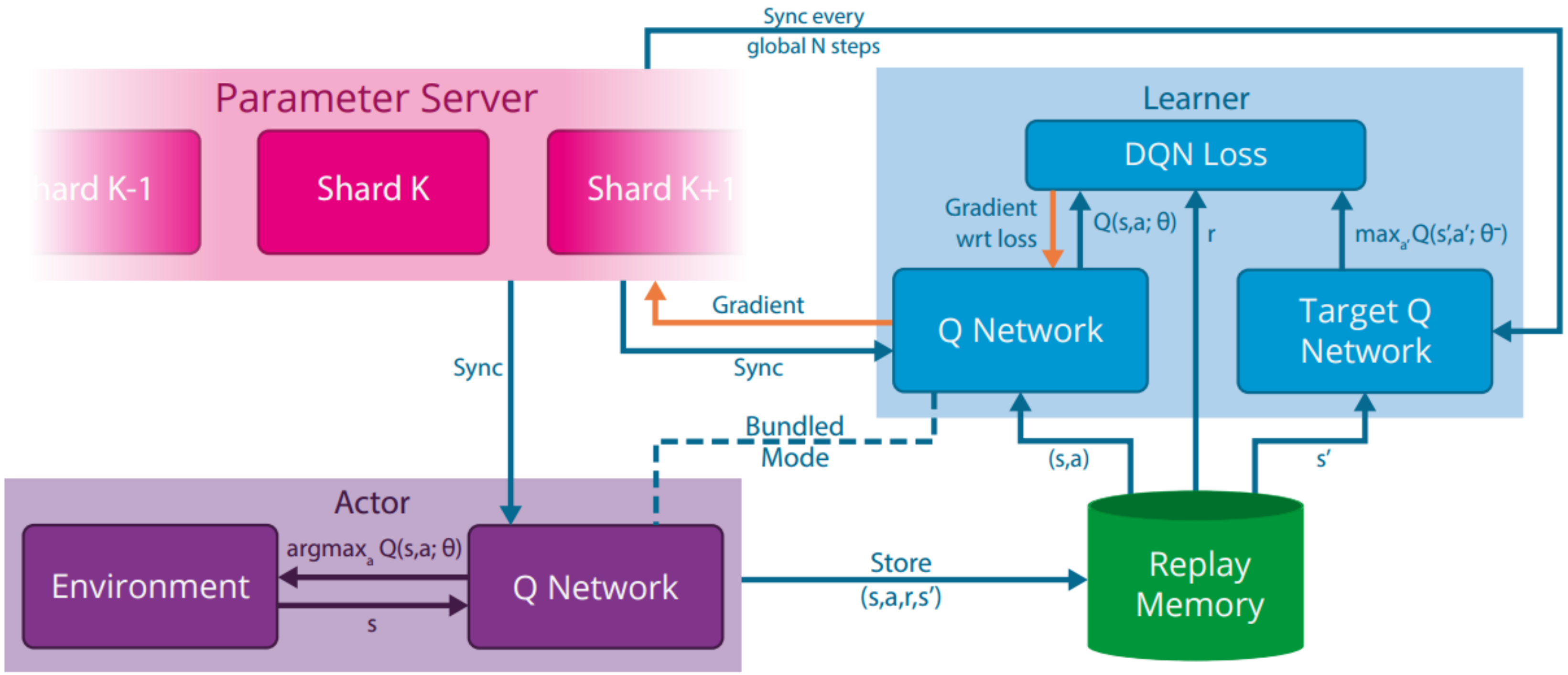


Results



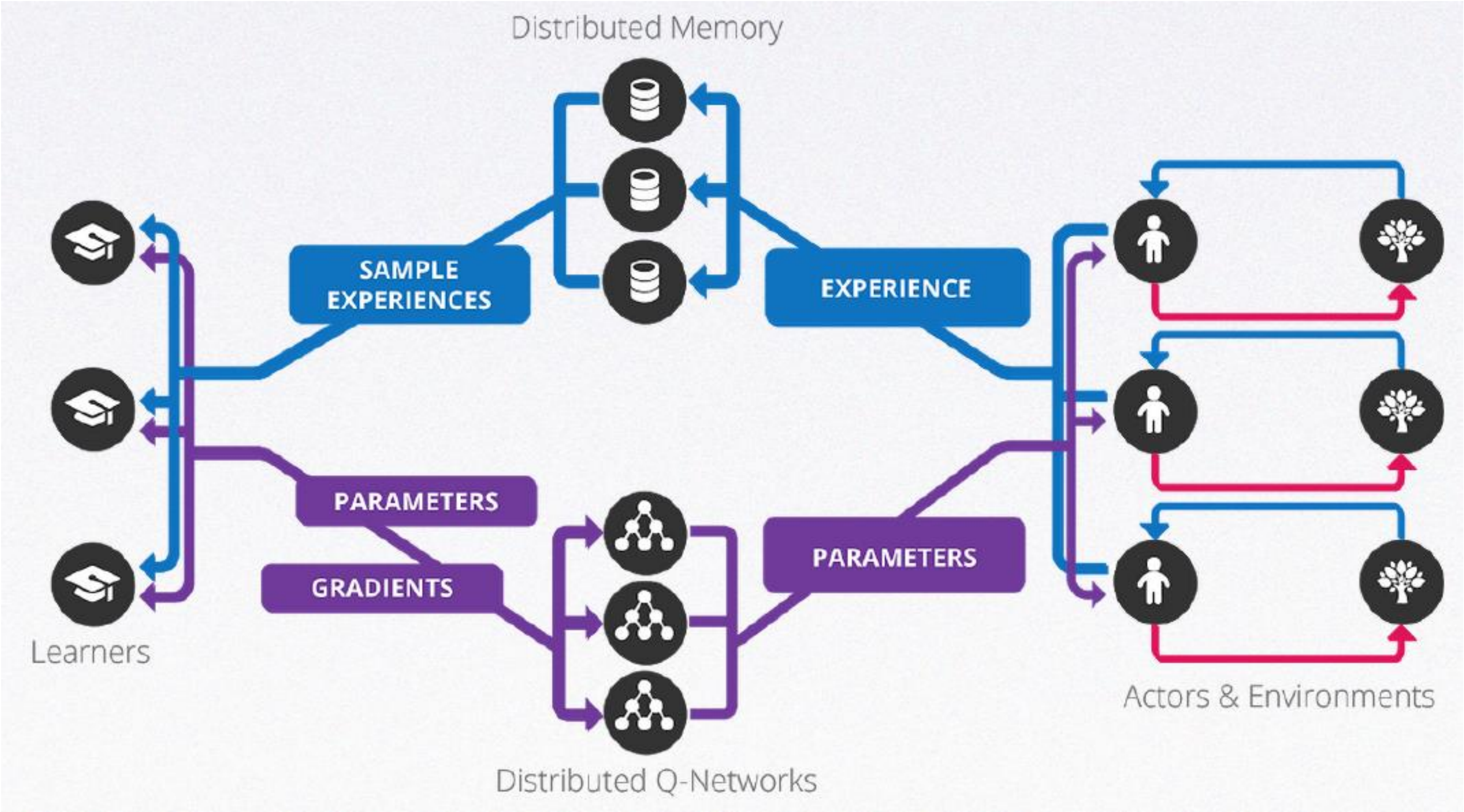
DQN
Best Linear Learner
at human-level or above
below human-level
3000%
1000%
600%
500%
400%
300%
200%
100%
0%
Video Pinball 2539%
Boxing 1707%
Breakout 1327%
Star Gunner 598%
Robotank 508%
Atlantis 451%
Crazy Climber 419%
Gopher 400%
Demon Attack 294%
Name This Game 278%
Krull 277%
Assault 246%
Road Runner 232%
Kangaroo 224%
Tennis 148%
James Bond 145%
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%
Battle Zone 67%
Wizard of Wor 67%
Chopper Command 64%
Centipede 62%
Bank Heist 57%
River Raid 57%
Amidar 43%
Alien 42%
Venture 32%
Seaquest 25%
Double Dunk 17%
Bowling 14%
Ms. Pacman 13%
Asteroids 7%
Frostbite 6%
Gravitar 5%
Private Eye 2%
Montezuma's Revenge 0%
336%
1045%
14%
4%
273%
308%
51%
102%
11%
11%
222%
31%
0%
52%
159%
86%
10%
7%
25%
65%
128%
66%
7%
51%
6%
35%
66%
5%
25%

Gorila



- Parallel acting: generate new interactions
- Distributed replay memory: save interactions
- Parallel learning: compute gradients from replayed interactions
- Distributed neural network: update network from gradients

Gorila



Distributed Memory
SAMPLE EXPERIENCES
EXPERIENCE
PARAMETERS
GRADIENTS
PARAMETERS
Learners
Actors & Environments
Distributed Q-Networks

Policy Gradient

- Define the loss function of policy $\pi(*: \mu)$ as

$$\mathcal{H}(\mu) = \mathbb{E}[r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots | \pi(*: \mu)]$$

- For stochastic policy $\pi(a|s: \mu)$

$$\frac{\partial \mathcal{H}(\mu)}{\partial \mu} = \mathbb{E}\left[Q^\pi(s, a) \frac{\partial \log \pi(a|s: \mu)}{\partial \mu}\right]$$

- For deterministic policy $a = \pi(s: \mu)$ when a is continuous and Q is differentiable

$$\frac{\partial \mathcal{H}(\mu)}{\partial \mu} = \mathbb{E}\left[\frac{\partial Q^\pi(s, a)}{\partial a} \frac{\partial a}{\partial \mu}\right]$$

Deterministic Actor-Critic

Horizon Robotics

- A critic network estimates value of current policy by Q-learning

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta} = \mathbb{E}\left[\left(r + \gamma Q(s', \pi(s'): \theta) - Q(s, a: \theta)\right) \frac{\partial Q(s, a: \theta)}{\partial \theta}\right]$$

- An actor network updates policy in direction that improves Q

$$\frac{\partial \mathcal{H}(\mu)}{\partial \mu} = \mathbb{E}\left[\frac{\partial Q(s, a: \theta)}{\partial a} \frac{\partial a}{\partial \mu}\right]$$

- However, naive actor-critic oscillates or diverges with deep neural nets.

Deep Deterministic Actor-Critic

Horizon Robotics

- Use experience replay for both actor and critic
- Use target Q-network to avoid oscillations

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta} = \mathbb{E}_{s,a,r,s'\sim\mathcal{D}}\left[\left(r + \gamma Q(s', \pi(s'): \bar{\theta}) - Q(s, a: \theta)\right)\frac{\partial Q(s, a: \theta)}{\partial \theta}\right]$$

$$\frac{\partial \mathcal{H}(\mu)}{\partial \mu} = \mathbb{E}_{s,a,r,s'\sim\mathcal{D}}\left[\frac{\partial Q(s, a: \theta)}{\partial a}\frac{\partial a}{\partial \mu}\right]$$

- These changes provide much more stable solution.

Future of RL

Horizon Robotics

- Learning from experts: interactive learning
- State space exploration: Monte Carlo methods, curiosity driven, incentive driven
- Transition model: GAN (cGAN, wGAN, etc.)
- Auxiliary/Multi tasks framework: scene parsing, depth estimation, flow estimation, etc.
- More “tricks”: prioritized experience replay, noise model
- Virtuality & reality: switch between simulator and hardware
- …

Horizon Robotics

Welcome to join us!

dream@hobot.cc