Machine Learning (part II)

Reinforcement Learning

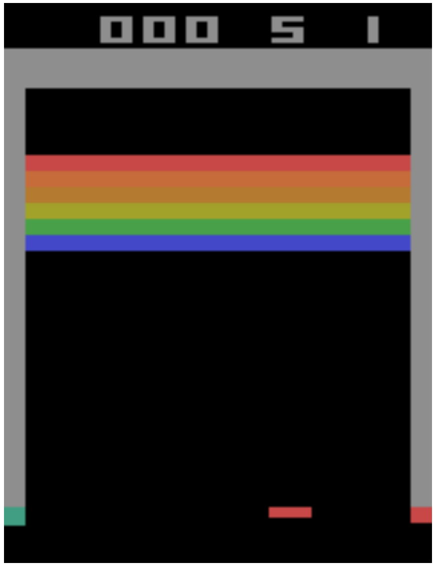
Angelo Ciaramella

Università Parthenope
DiST
Dipartimento di Scienze e Tecnologie

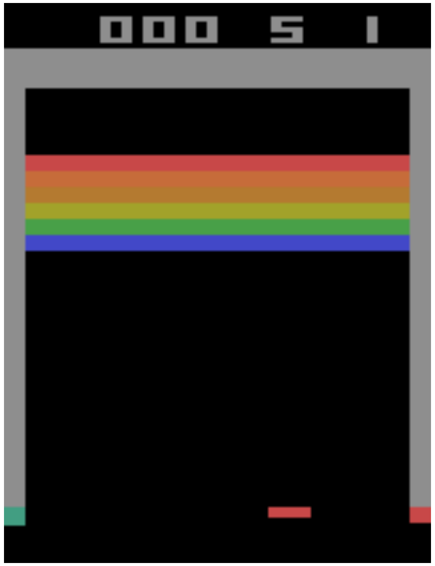
What is Reinforcement Learning?

- Learning from interaction with an environment
  - to achieve some long-term goal that is related to the state of the environment
- The goal is defined by reward signal, which must be maximised
- Agent must be able to partially/fully sense the environment state and take actions to influence the environment state
- The state is typically described with a feature-vector

RL Demo



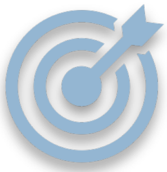
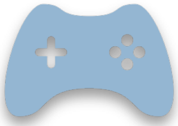
Random



DQN

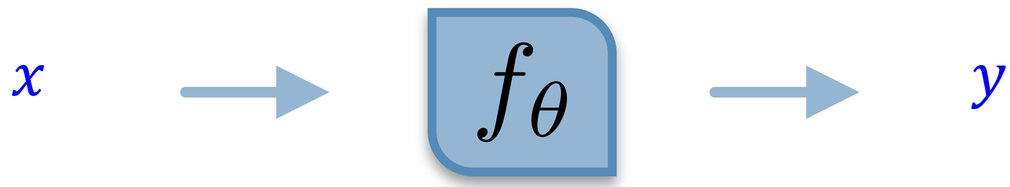
Atari game

Learning approaches

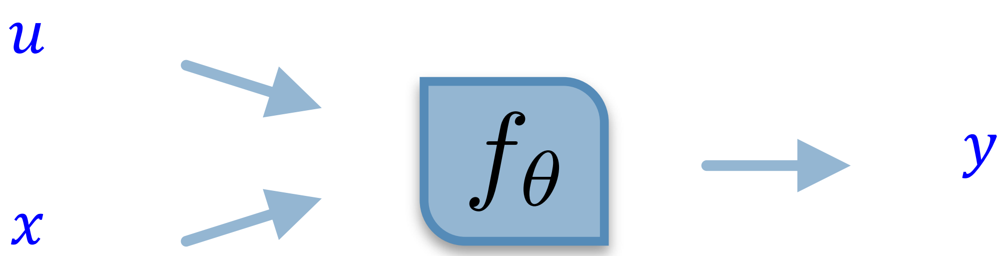
			
Supervised Learning	Unsupervised Learning	Reinforcement Learning	Paradigm
$p_\theta(y\|x)$	$p_\theta(x)$	$\pi_\theta(a\|s)$	Objective
→ Classification → Regression	→ Inference → Generation	→ Prediction → Control	Applications

Prediction vs Control

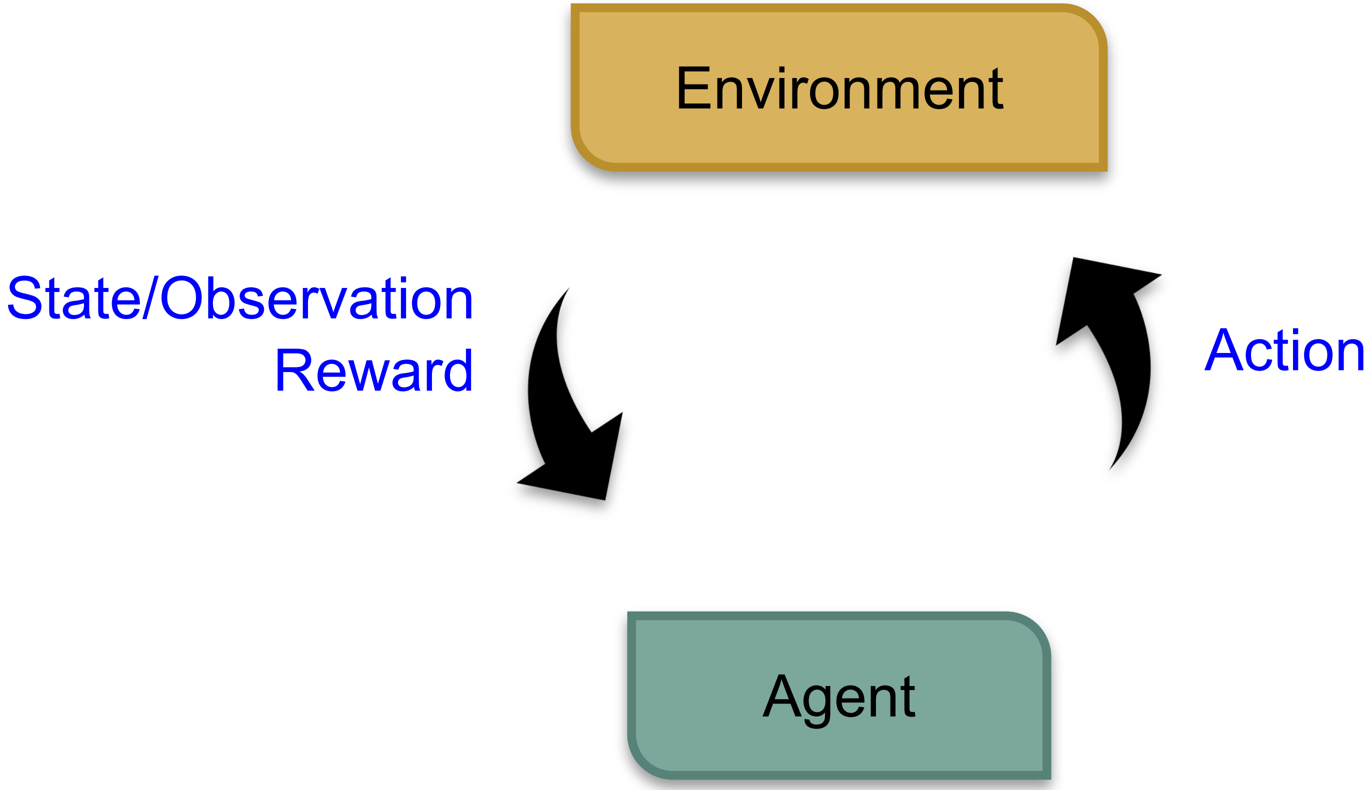
Prediction



Control

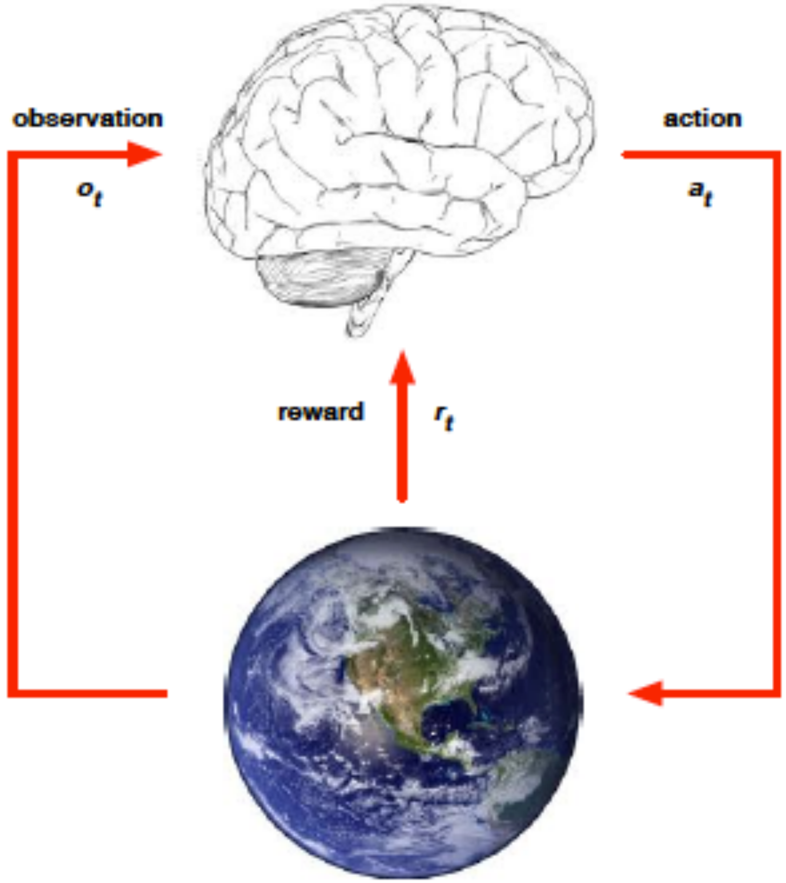


Setting

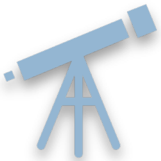


using policy $\pi_\theta(a|s)$

Agent and environmet

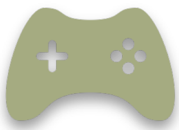


Markov Decision Process (MDP)



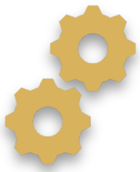
State space

$$s_t \in \mathcal{S}$$



Action space

$$a_t \in \mathcal{A}$$



Transition function

$$\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$$

$$s_{t+1} \sim \mathcal{T}(\cdot | s_t, a_t)$$

$$s_0 \sim \mathcal{T}_0$$

Reward function

$$\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$$

$$r_t \sim \mathcal{R}(s_t, a_t)$$

Discount factor

- We want to be greedy but not impulsive

- Implicitly takes uncertainty in dynamics into account

- Mathematically
  - $\gamma<1$ allows infinite horizon returns

$$G(s_t, a_t) = \sum_{\tau=0}^{T} \gamma^{\tau} \mathcal{R}(s_{t+\tau}, a_{t+\tau})$$

Solving an MDP

- Objective

$$J(\pi) = \mathbb{E}_{a_t \sim \pi(\cdot|s_t), s_{t+1} \sim \mathcal{T}(\cdot|s_t, a_t), s_0 \sim \mathcal{T}_0} \left[\sum_{t=0}^{T} \gamma^t \mathcal{R}(s_t, a_t) \right]$$

$$\hat{\pi} = \arg\max_{\pi} J(\pi)$$

State

- Experience
  - sequence of observations, actions, rewards

$$o_1, r_1, a_1, ..., a_{t-1}, o_t, r_t$$

- State
  - summary of experience

$$s_t = f(o_1, r_1, a_1, ..., a_{t-1}, o_t, r_t)$$

  - In a fully observed environment

$$s_t = f(o_t)$$

Major Components

- An RL agent may include one or more of these components
  - Policy
    - Agent's behaviour function
  - Value function
    - how good is each state and/or action
  - Model
    - Agent's representation of the environment

Policy

- Policy
  - agent's behaviour
  - It is a map from state to action
- Deterministic policy

$$a = \pi(s)$$

- Stochastic policy

$$\pi(a|s) = \mathbb{P}[a|s]$$

Value function

- A value function
  - prediction of future reward
    - *How much reward will I get from action a in state s ?*

- Q-value function
  - expected total reward
    - from state s and action a
    - under policy π
    - with discount factor γ

$$Q^{\pi}(s,a) = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \mid s, a\right]$$

Bellman equation

- Value functions decompose into a Bellman equation

$$Q^{\pi}(s,a) = \mathbb{E}_{s',a'}\left[r + \gamma Q^{\pi}(s',a') \mid s,a\right]$$

- Optimal value

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a) = Q^{\pi^{*}}(s,a)$$

$$\pi^{*}(s) = \operatorname*{argmax}_{a} Q^{*}(s,a)$$

Optimal value

- Optimal value maximises over all decisions

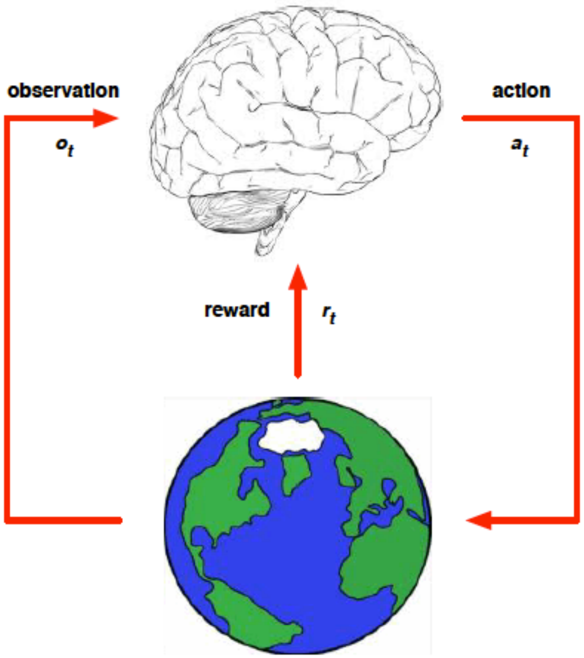
$$
\begin{aligned}
Q^*(s,a) &= r_{t+1} + \gamma \max_{a_{t+1}} r_{t+2} + \gamma^2 \max_{a_{t+2}} r_{t+3} + \ldots \\
&= r_{t+1} + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1})
\end{aligned}
$$

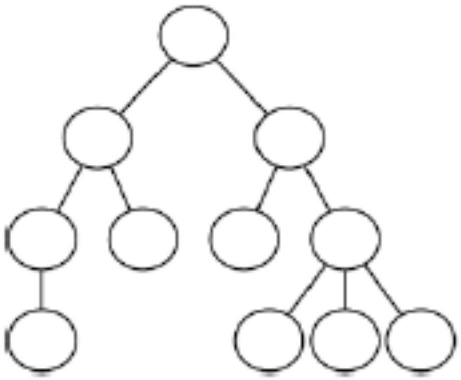
- Formally

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

Model

- Model
  - learnt from experience
  - acts as proxy for environment
  - planner interacts with model
  - e.g. using lookahead search





Approaches to Reinforcement Learning

- 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

Approsches to RL

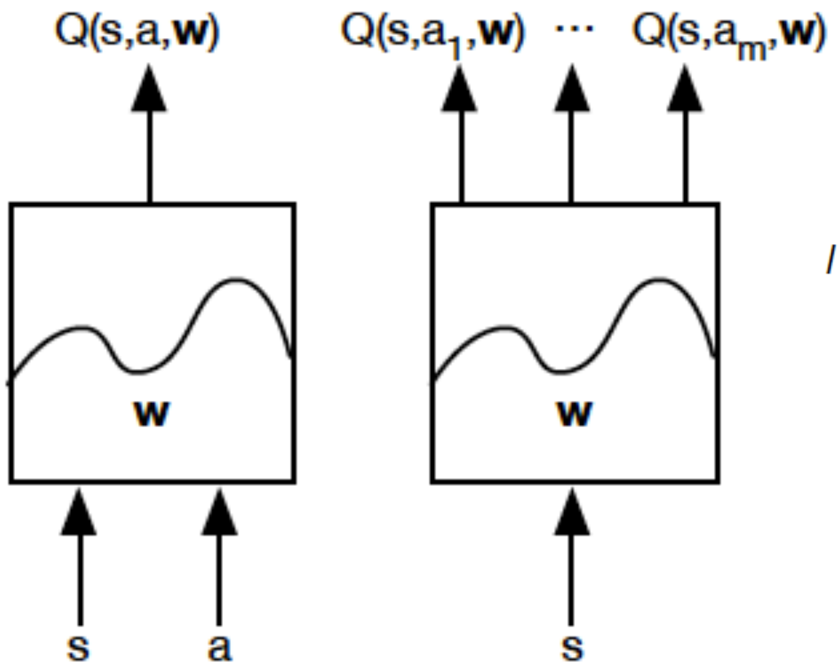
- Model-based RL
  - Build a model of the environment
  - Plan (e.g. by lookahead) using model
- Deep RL
  - Use deep neural networks to represent
    - Value function
    - Policy
    - Model
  - Optimise loss function by stochastic gradient descent

VB RL: Q-Networks

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

Bellman equation

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



$$l = \left(r + \gamma \max_{a} Q(s',a',\mathbf{w}) - Q(s,a,\mathbf{w})\right)^2$$

MSE loss

VB RL: Q-learning

- Optimal Q-values should obey Bellman equation

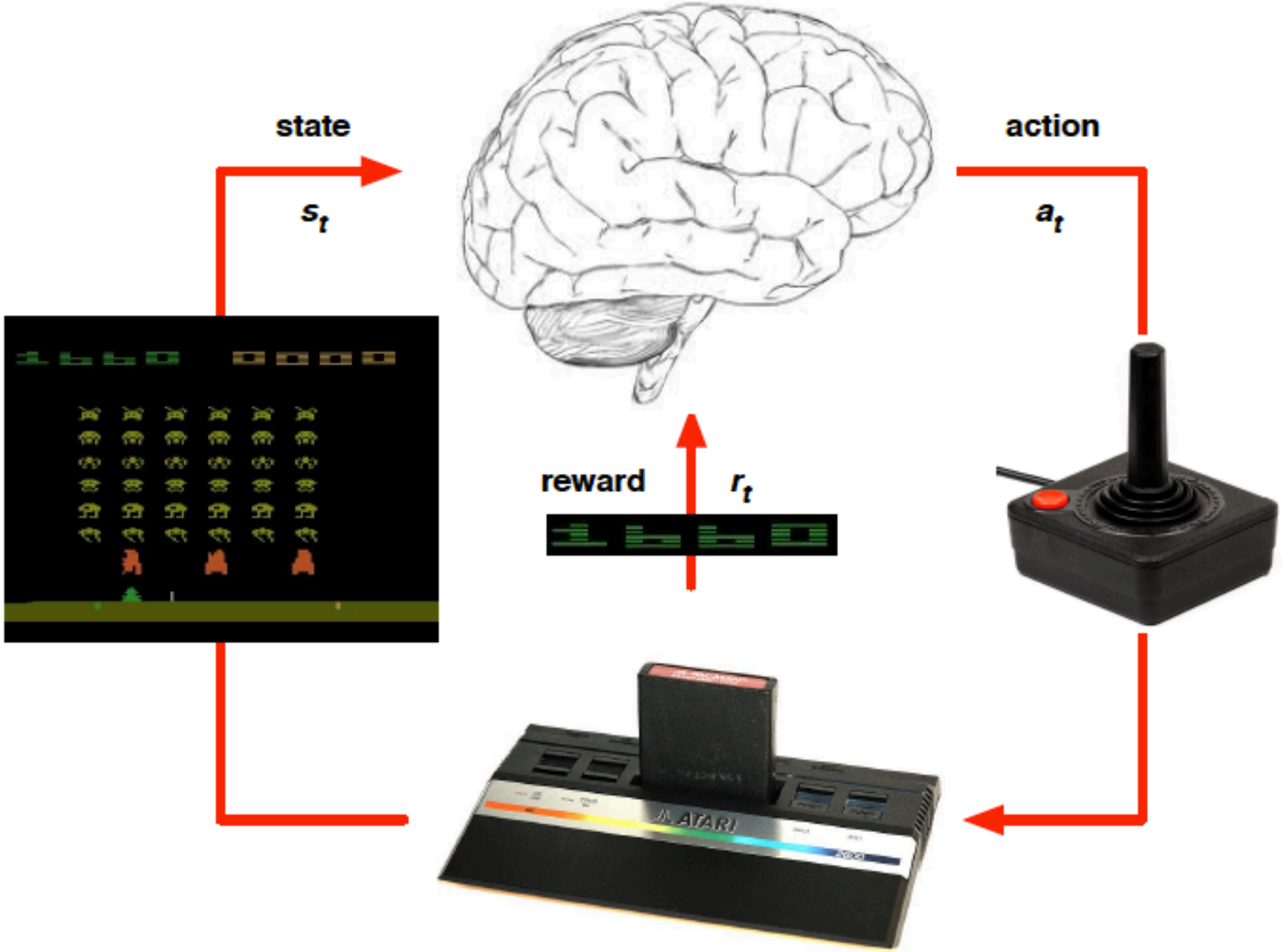
$$Q^*(s, a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q(s', a')^* \mid s, a\right]$$

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

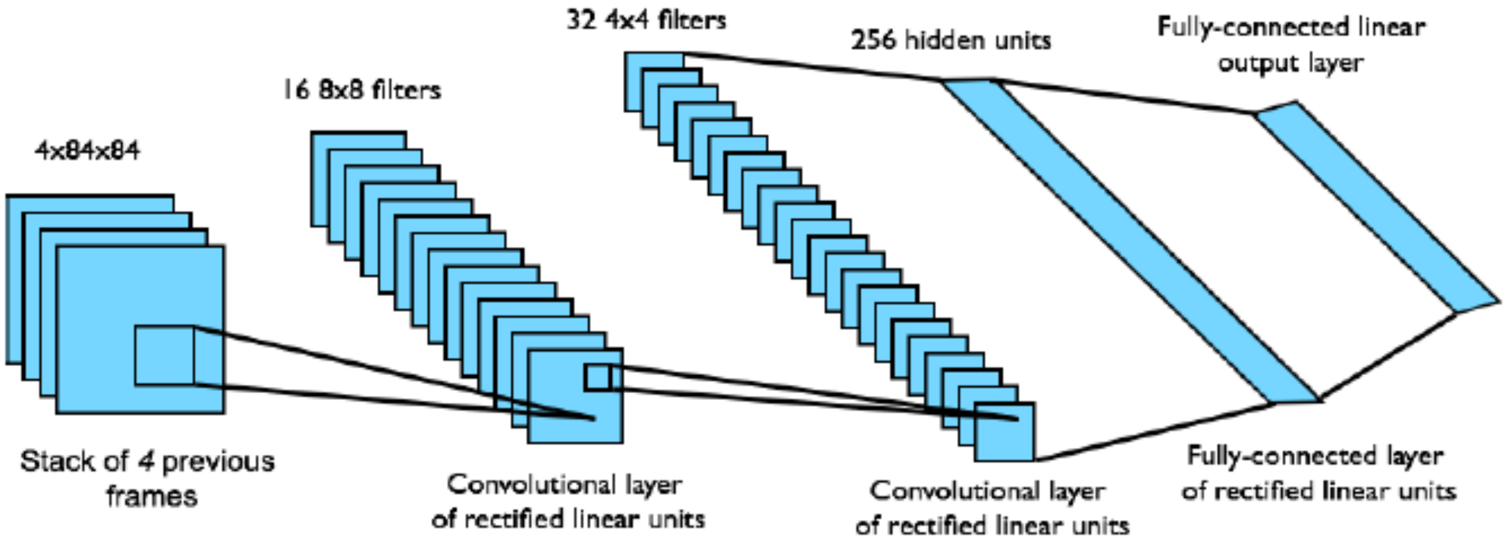
$$l = \left(r + \gamma \max_{a} Q(s', a', \mathbf{w}) - Q(s, a, \mathbf{w})\right)^2$$

MSE

VB RL: DQN



VB RL: DQN in Atari



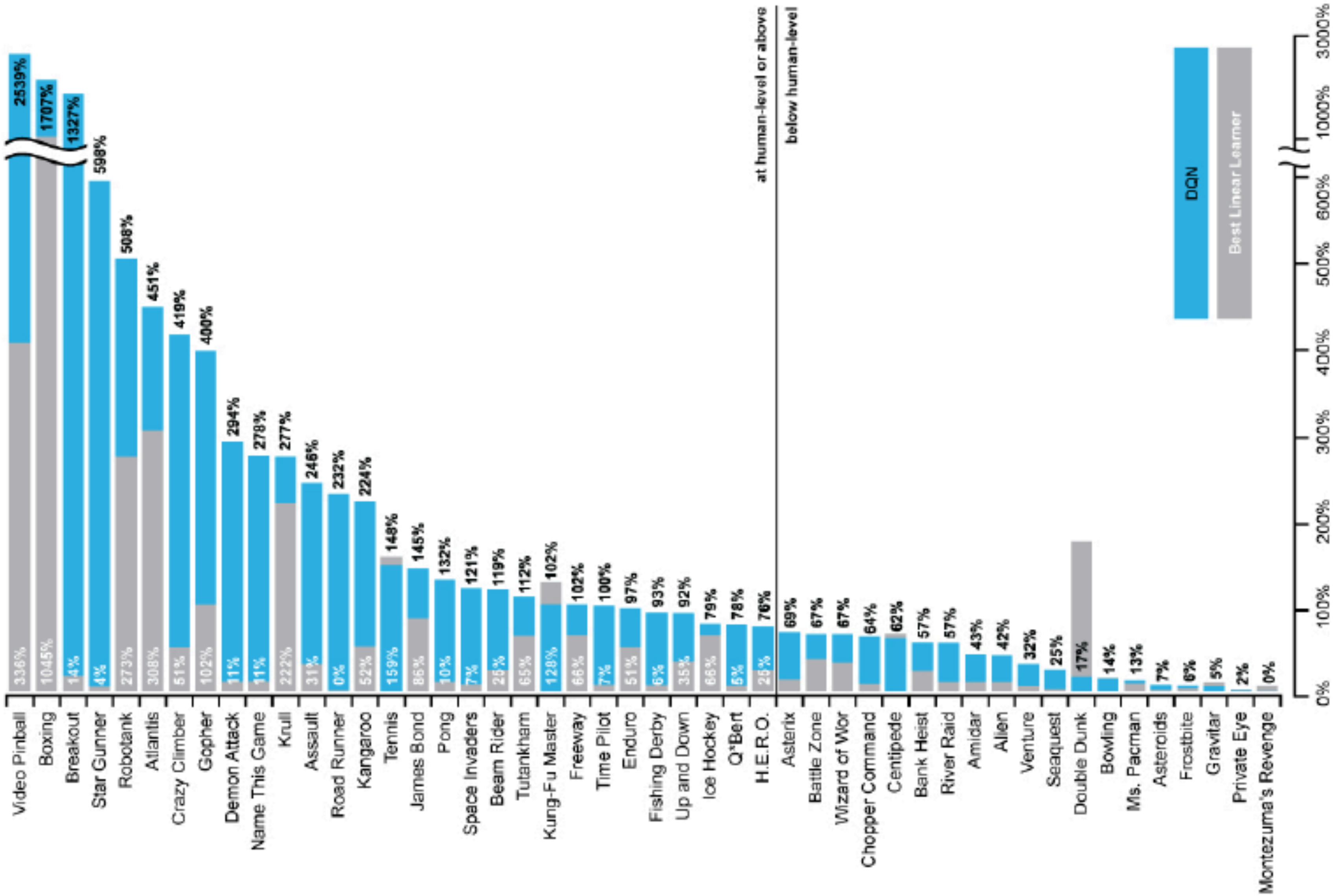
End-to-end learning of values $Q(s, a)$ from pixels s

Input state s is stack of raw pixels from last 4 frames

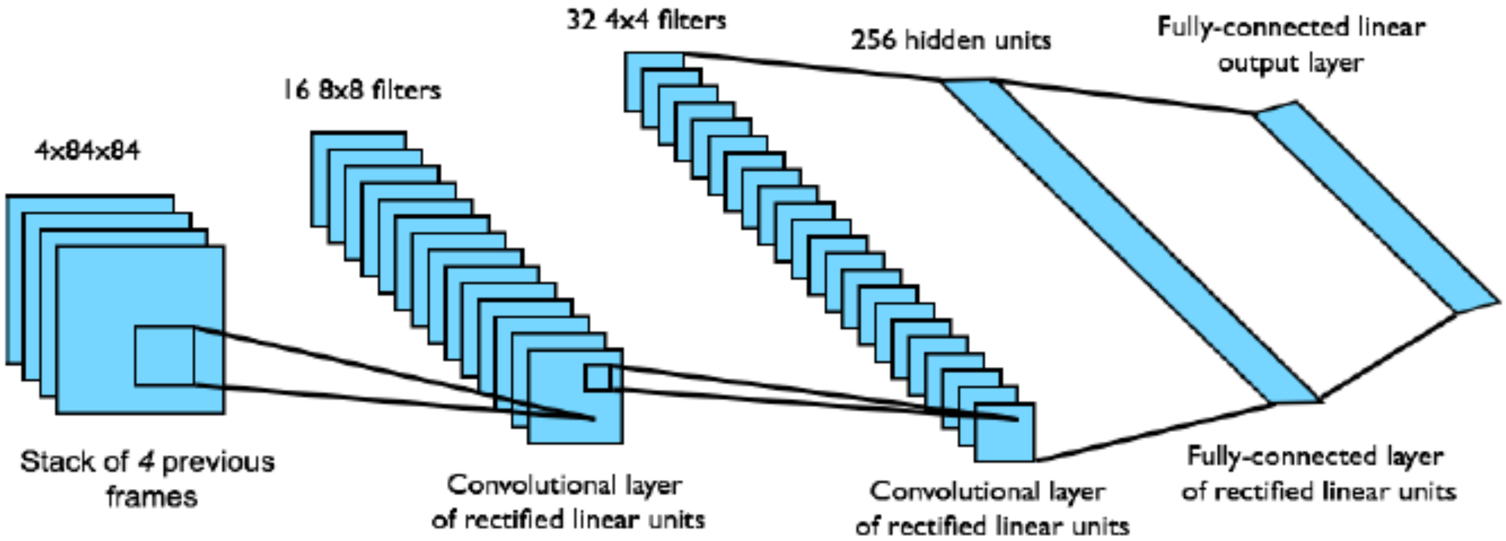
Output is $Q(s, a)$ for 18 joystick/button positions

Reward is change in score for that step

VB RL: DQN results in Atari



VB RL: DQN 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

VB RL: DQN results in Atari

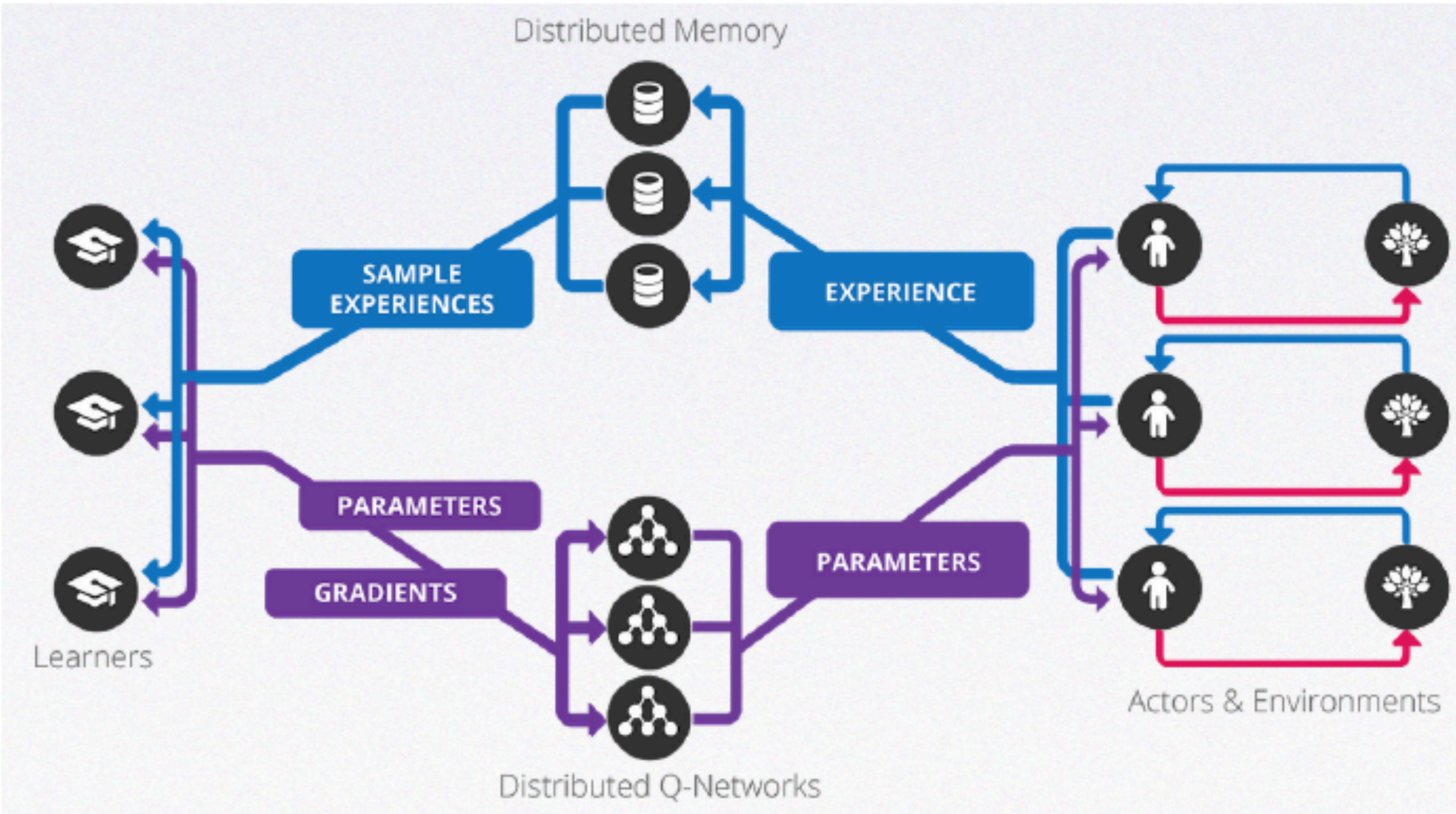
DQN paper

www.nature.com/articles/nature14236

DQN source code:

sites.google.com/a/deepmind.com/dqn/

Gorila (General Reinforcment Learning Architecture)



Deep Policy Networks

- Represent policy by deep network with weights u

$$a = \pi(a|s, \mathbf{u}) \text{ or } a = \pi(s, \mathbf{u})$$

- Objective function as total discounted reward

$$L(\mathbf{u}) = \mathbb{E}\left[r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots \mid \pi(\cdot, \mathbf{u})\right]$$

- Optimise
  - objective end-to-end by SGD
    - i.e. Adjust policy parameters u to achieve more reward

Deep Policy Networks

- The gradient of a stochastic policy

$$\frac{\partial L(\mathbf{u})}{\partial u} = \mathbb{E}\left[\frac{\partial \log \pi(a|s,\mathbf{u})}{\partial \mathbf{u}} Q^{\pi}(s,a)\right]$$

$$\frac{\partial L(\mathbf{u})}{\partial \mathbf{u}} = \mathbb{E}\left[\frac{\partial Q^{\pi}(s,a)}{\partial a}\frac{\partial a}{\partial \mathbf{u}}\right] \qquad a = \pi(s)$$

Actor-Critic Algorithm

Model-Based Deep RL

- Learning Models of the Environment
  - Challenging to plan due to compounding errors
    - Errors in the transition model compound over the trajectory
    - Planning trajectories differ from executed trajectories
    - At end of long, unusual trajectory, rewards are totally wrong

DRL in Go

AlphaGo paper:

www.nature.com/articles/nature16961

AlphaGo resources:

deepmind.com/alphago/