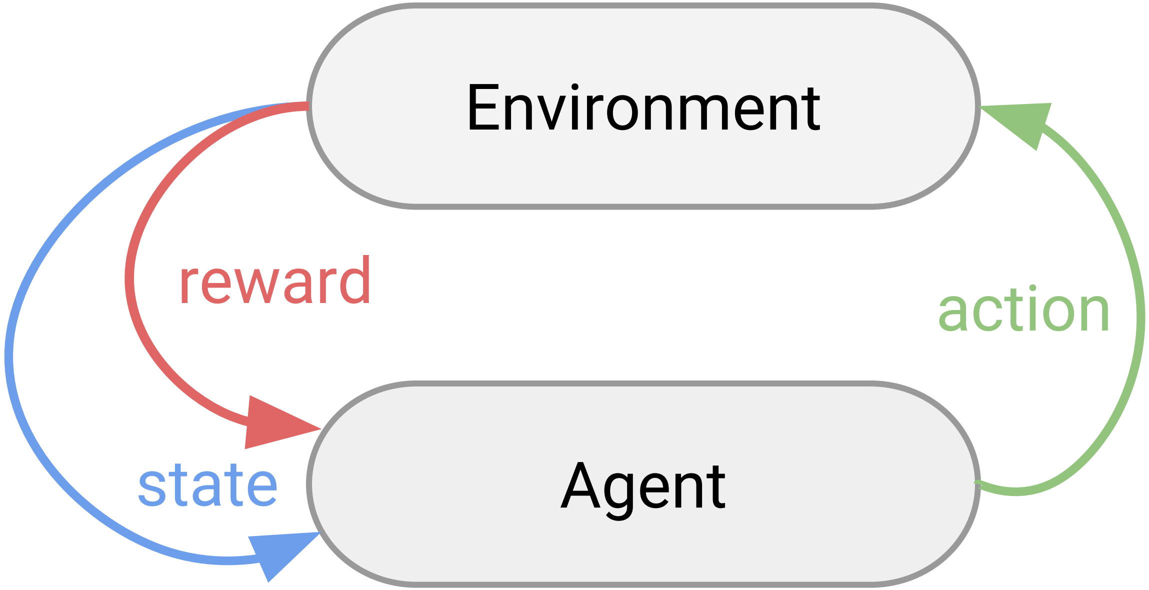
Learning to Do

Supervised Learning: given **data,**
predict **labels**

Unsupervised Learning: given **data**,
learn about that **data**

Reinforcement Learning: given **data**,
choose **action** to maximize expected
long-term reward


10 #
Boston Dynamics

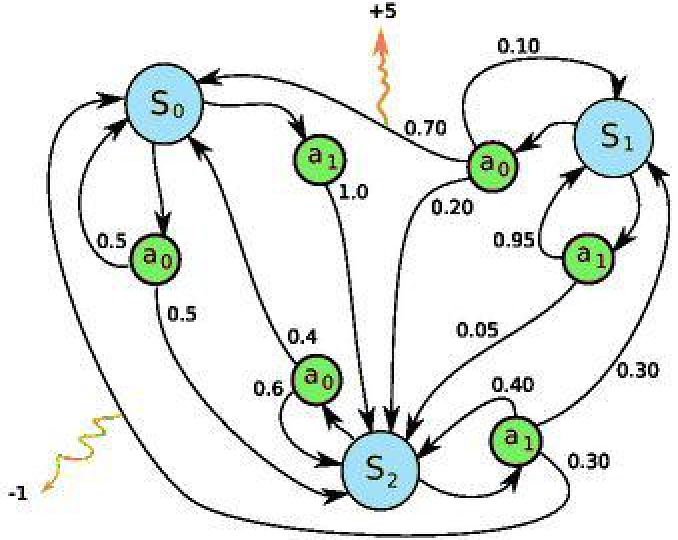

Environment
reward
action
state
Agent

Episode: sequence of states and actions

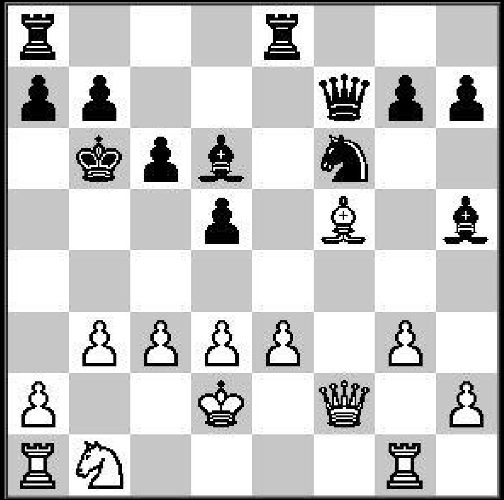
$$s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T, r_T$$

Transition function:

$$P(s_{t+1}, r_t \mid s_t, a_t)$$




Boston Dynamics



We want to find a policy $\pi(s) = p(a|s)$ which maximizes

$$\cancel{r_0 + r_1 + r_2 + \cdots + r_T}$$

$$\gamma OLO$$

$$\cancel{r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots + \gamma^T r_T}$$

$$E\left[r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots + \gamma^T r_T\right]$$

$$\max_{\pi} E\left[\sum_{i=0}^{T} \gamma^i r_i\right]$$

Policy Learning

$$\text{Find } \pi(s)$$

$$a \sim \pi(s)$$

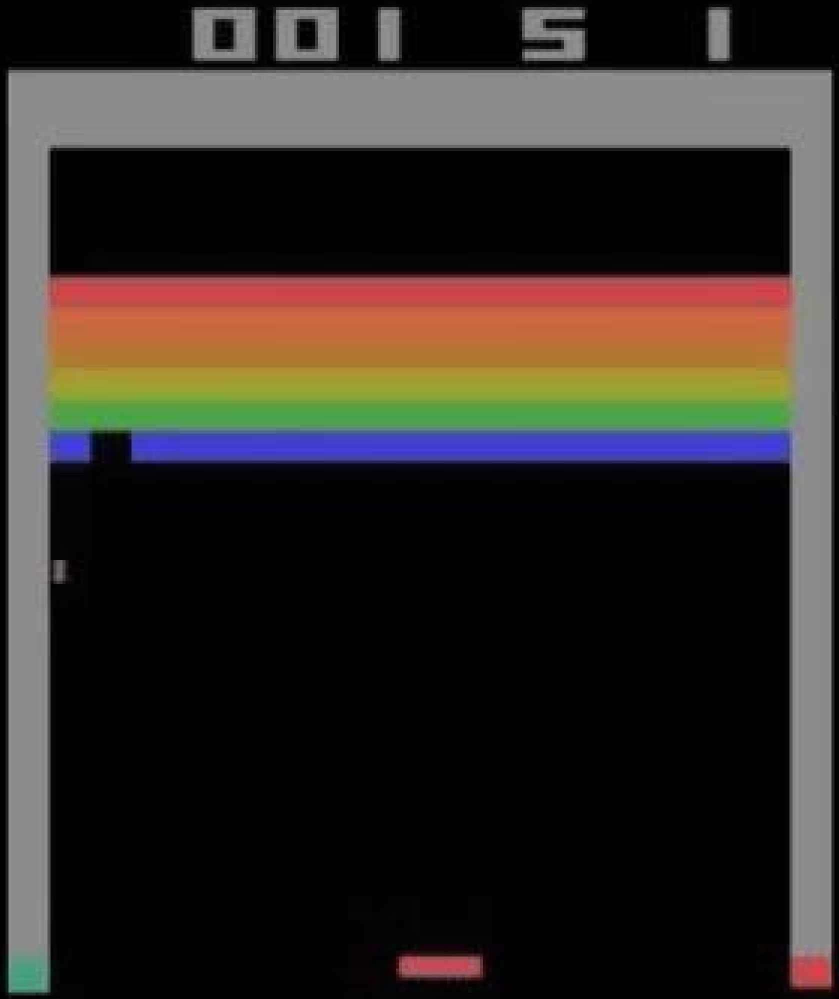
Value Learning

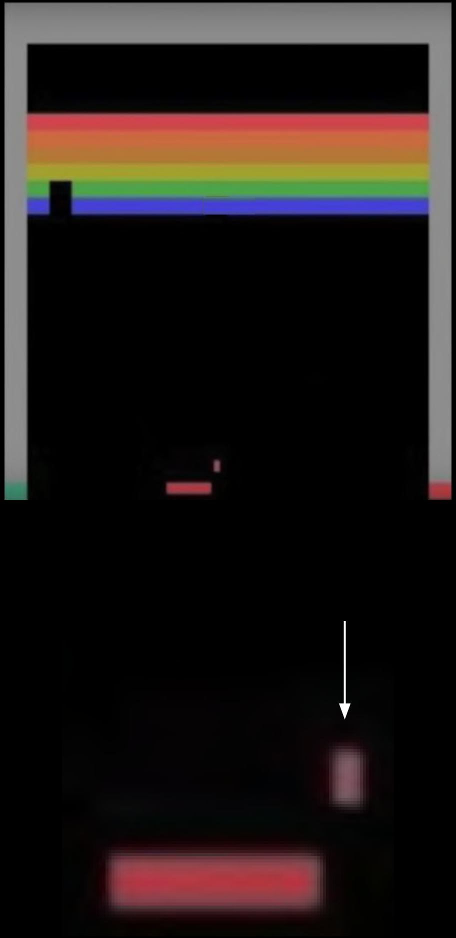
$$\text{Find } Q(s, a)$$

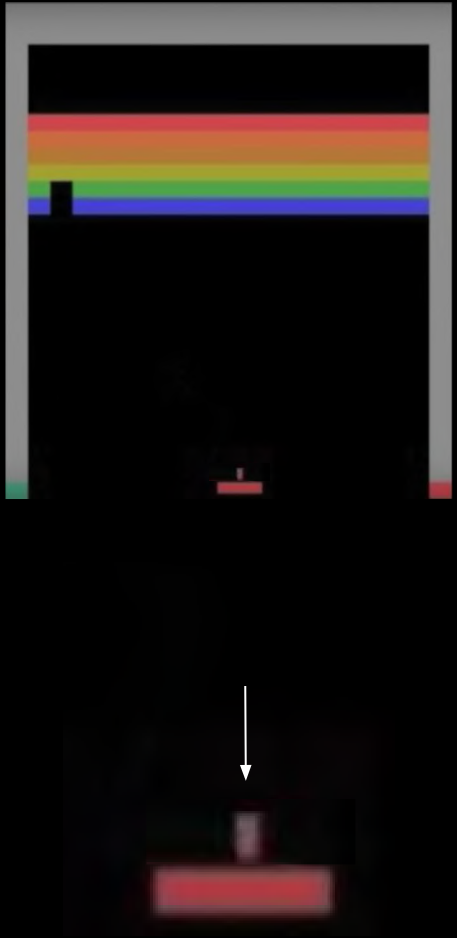
$$a = \underset{a'}{\arg\max}\, Q(s, a')$$

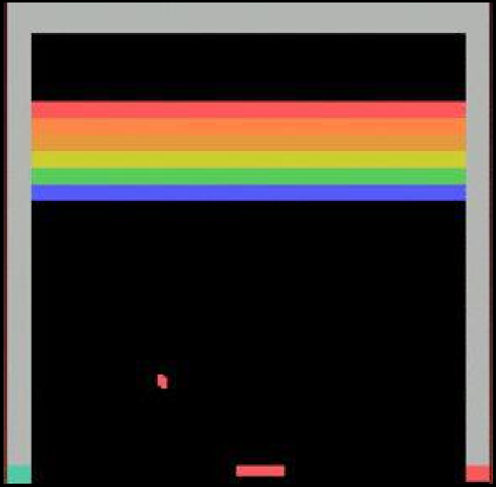
$$Q^*(s_t, a_t) = \max_\pi E\left[\sum_{i=t}^{T} \gamma^i r_i\right]$$

maximum expected future rewards starting at **state s_i**, choosing **action a_i** , and then following an **optimal policy** π^*


001 5 1







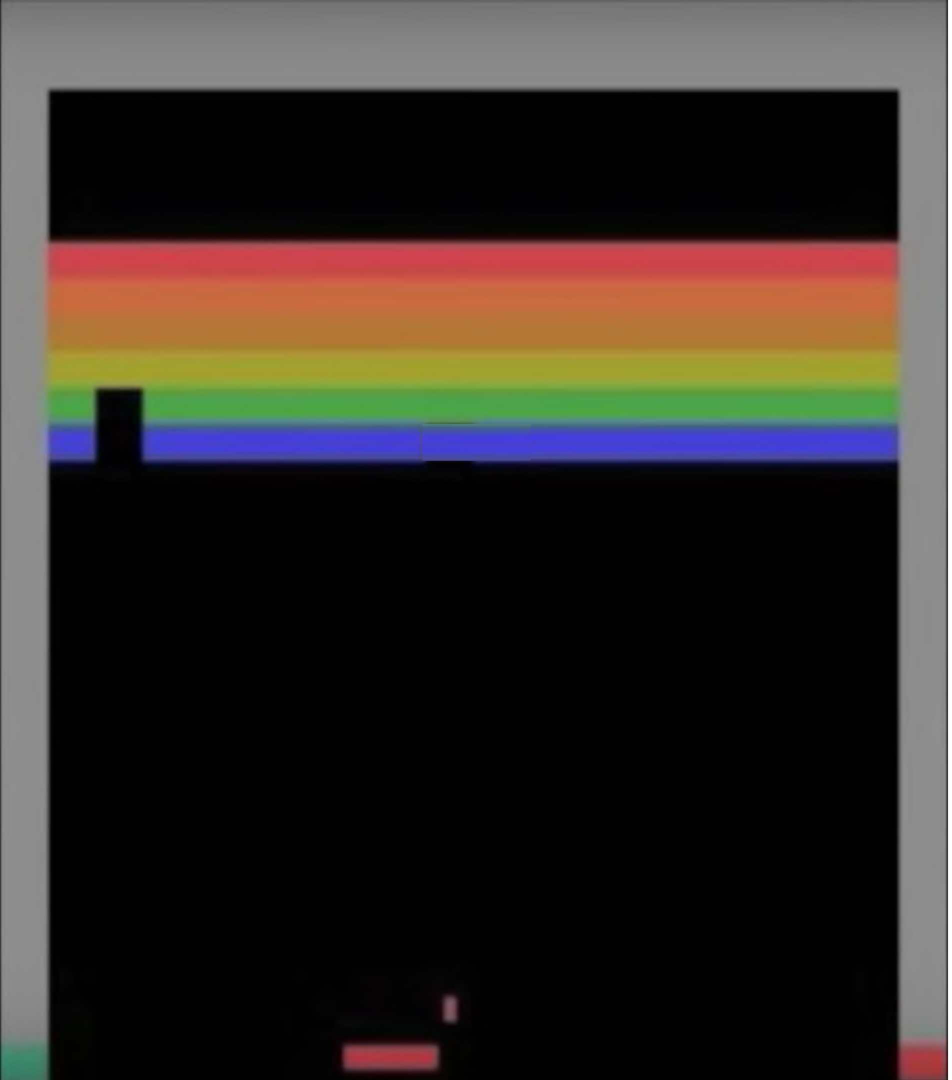
$$Q^*(s_t, a_t) = E\left[r_t + \gamma \max_{a'} Q^*(s_{t+1}, a')\right]$$

The **max future reward** for taking **action a_t** is the **current reward** plus the next step’s **max future reward** from taking the best next **action a’**

$$\widehat{Q_{j+1}}(s_t, a_t) \leftarrow E\left[r_t + \gamma \max_{a'} \widehat{Q_j}(s_{t+1}, a')\right]$$

$$\widehat{Q_j} \rightarrow \widehat{Q_{j+1}} \rightarrow \widehat{Q_{j+2}} \rightarrow \cdots \rightarrow Q^*$$

But… how large is $Q(\cdot)$?

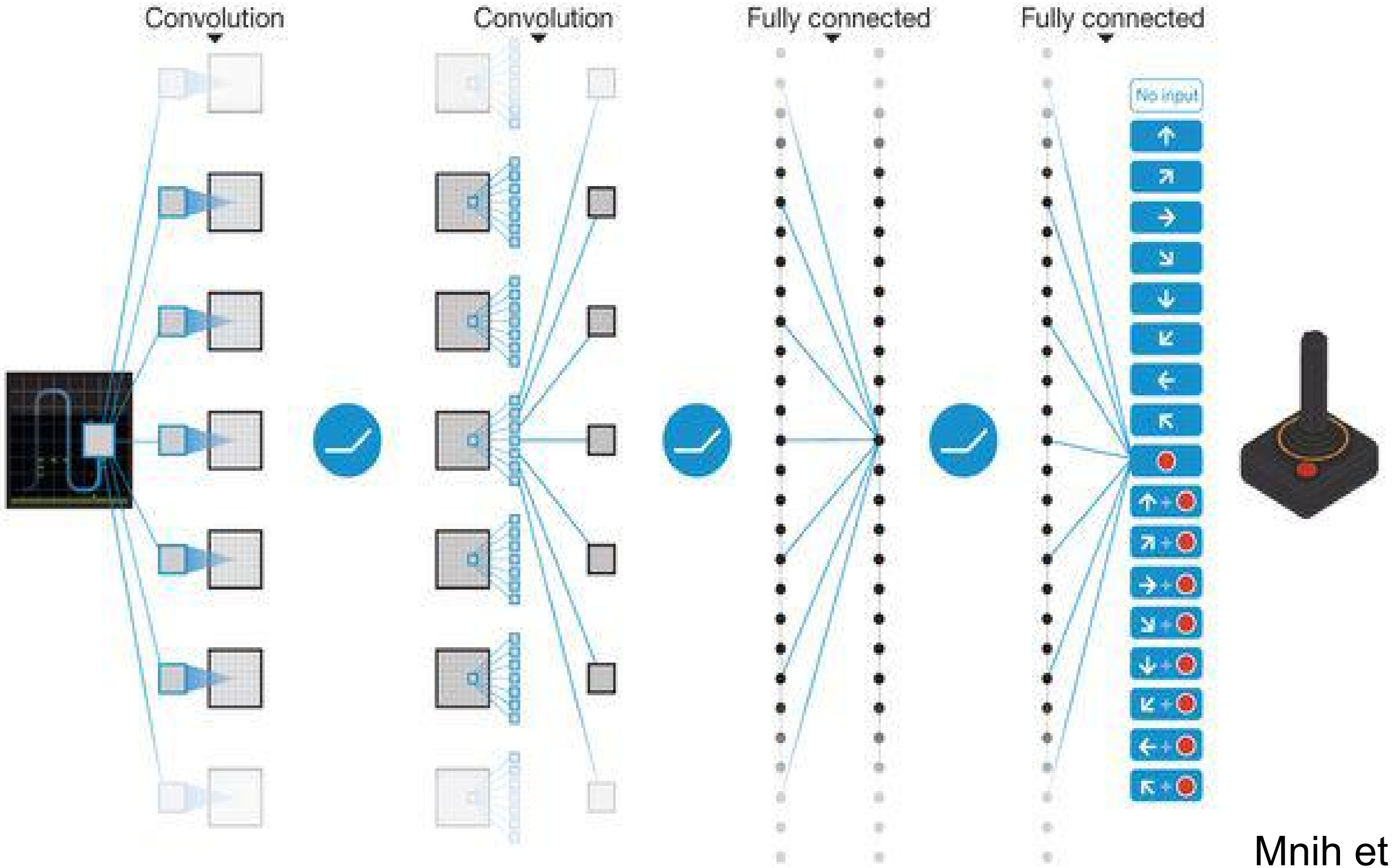


states: $\sim 2^{96} \cdot 60 \cdot 60$
actions: 3
Q values: $\sim 2^{111}$

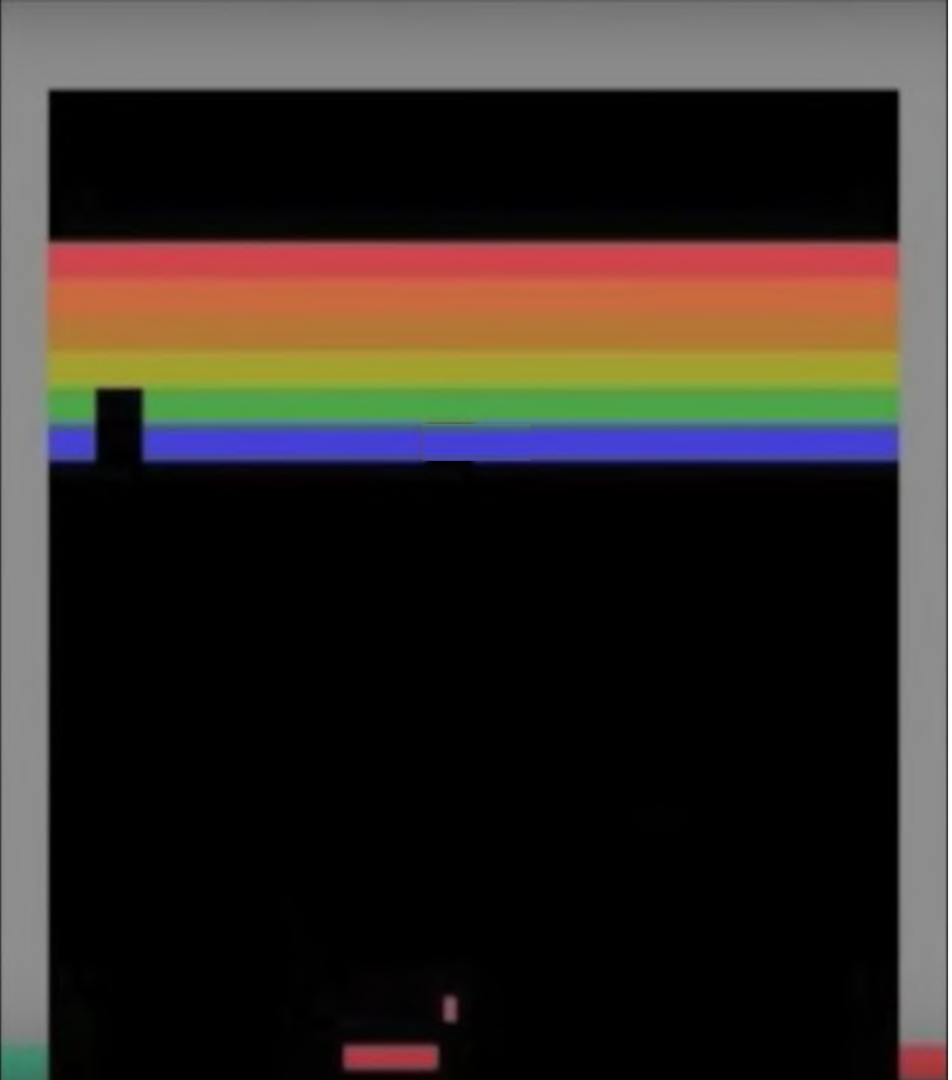
1957 - 2013

:(

ENTER THE DEEP



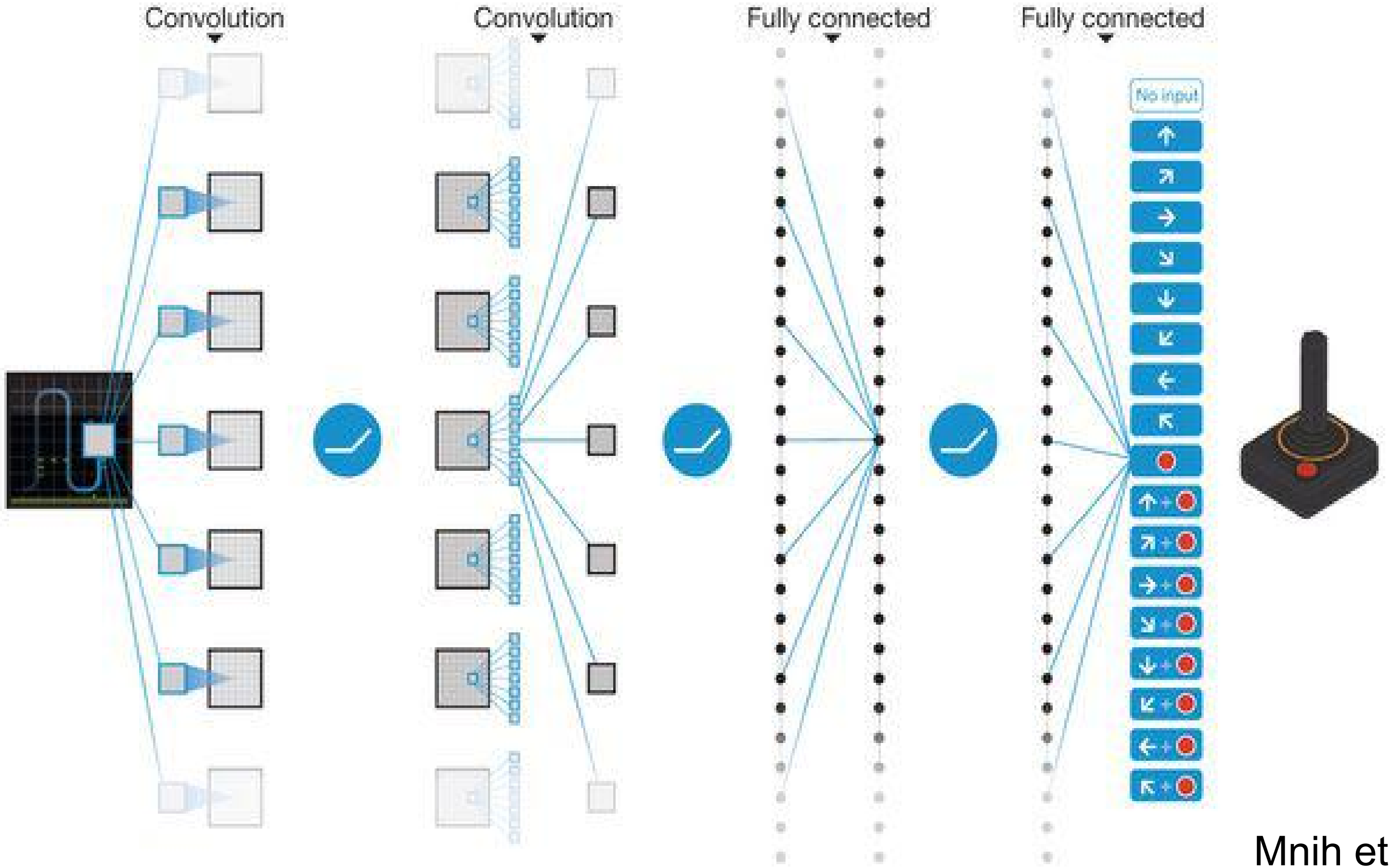
Mnih et al, 2013



Features for Estimating Q

- Whether the paddle can reach the ball
- # remaining blocks
- How are the blocks spatially arranged?

ENTER THE DEEP



Mnih et al, 2013

Define approximate Q* function

$$\widehat{Q_\theta}(s, a|\theta) \sim Q^*(s, a)$$

and choose θ to minimize

$$\min_{\theta} \sum_{e \in E} \sum_{t=0}^{T} \left\| \widehat{Q}(s_t, a_t|\theta) - \left(r_t + \gamma \max_{a'} \widehat{Q}(s_{t+1}, a'|\theta) \right) \right\|$$

function Q-LEARNING

  Initialize θ

  $s = s_0$

  while not bored yet **do**

    Choose a from some policy $\pi(s)$, and store results r, s_{new}

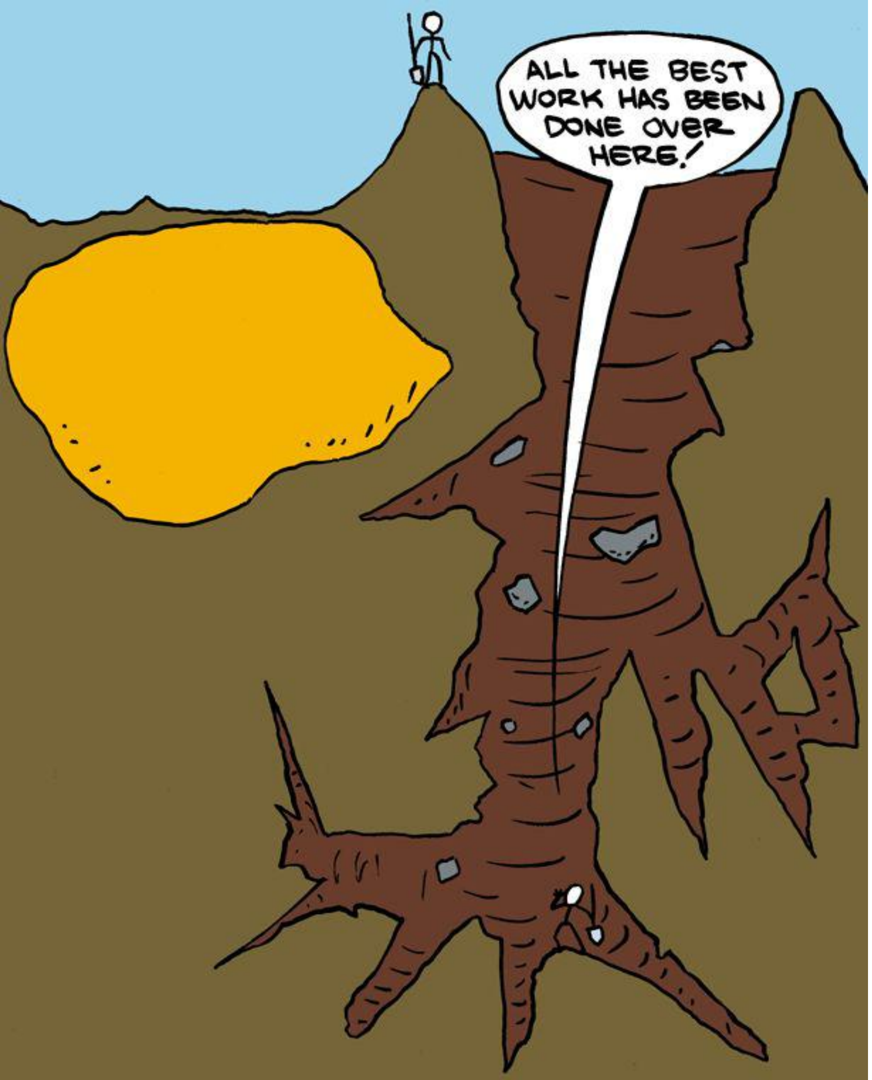
    Compute $\nabla_\theta E_Q = \nabla_\theta \left\| \widehat{Q}(s, a|\theta) - \left(r + \gamma \max_{a'} \widehat{Q}(s, a'|\theta_{old}) \right) \right\|$

    $\theta = \theta - \eta \nabla_\theta E_Q$

    $s = s_{new}$ (or s_0 if episode ended)

    $\theta_{old} = \theta$

function Q-LEARNING
    Initialize θ
    $s = s_0$
    while not bored yet **do**
        Choose a from some policy $\pi(s)$, and store results r, s_{new}
        Compute $\nabla_\theta E_Q = \nabla_\theta \left\| \widehat{Q}(s, a|\theta) - \left(r + \gamma \max_{a'} \widehat{Q}(s, a'|\theta_{old}) \right) \right\|$
        $\theta = \theta - \eta \nabla_\theta E_Q$
        $s = s_{new}$ (or s_0 if episode ended)
        $\theta_{old} = \theta$



We need to balance

exploration and **exploitation**

ϵ-greedy exploration

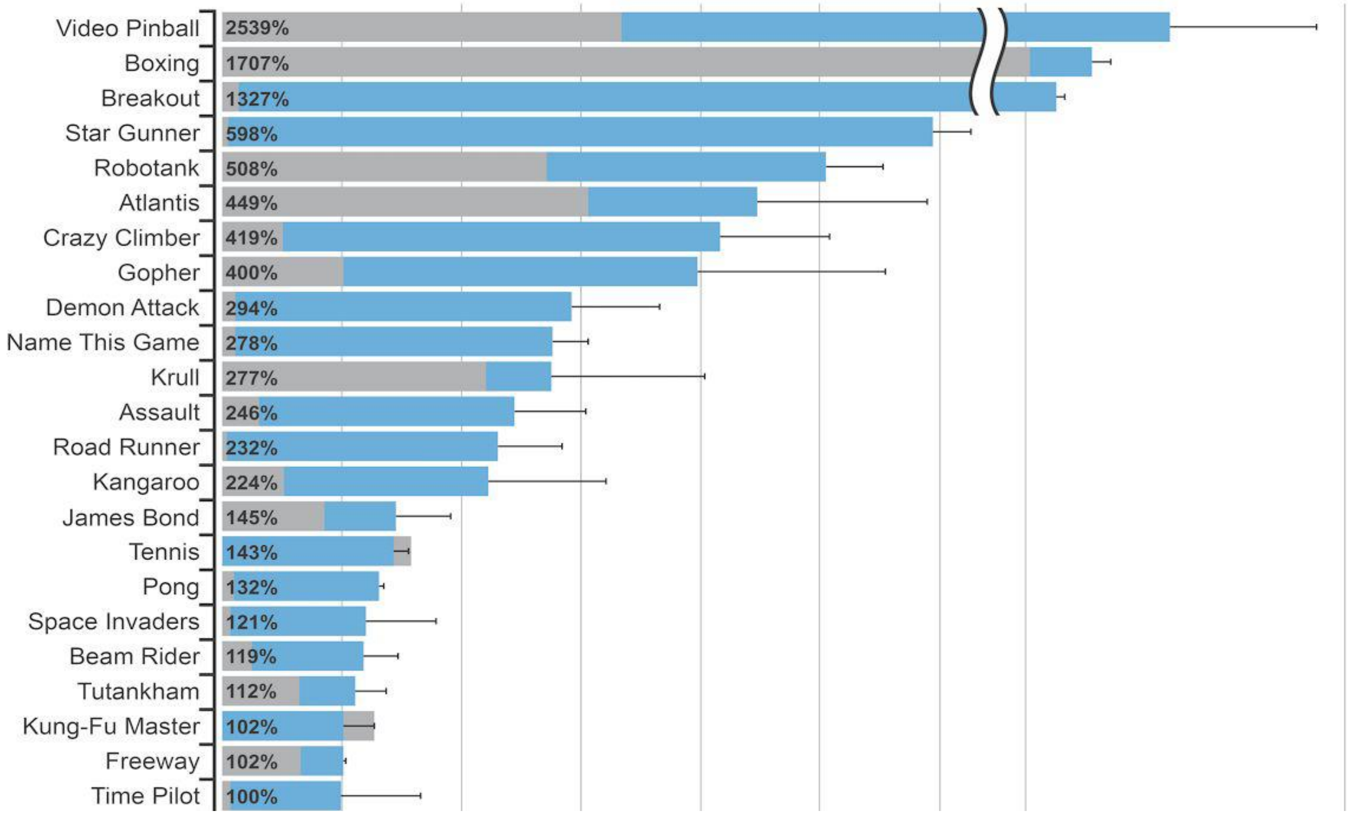
With probability $1-\epsilon$:

Pick $a_{t+1} \sim \operatorname{soft\,max}_{a'} \widehat{Q}(s_{t+1}, a')$

With probability ϵ :

Pick a_{t+1} at random


042 2 1



% improvement over professional player

Mnih et al, 2013

Try it out!

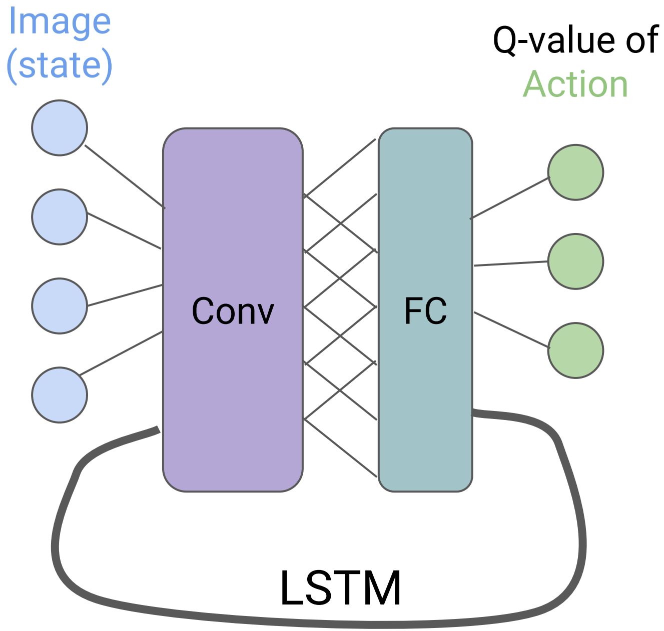
http://selfdrivingcars.mit.edu/deeptraffic/

(Kudos to Lex Fridman & the 6.S094 team!)

The problems with Q-learning

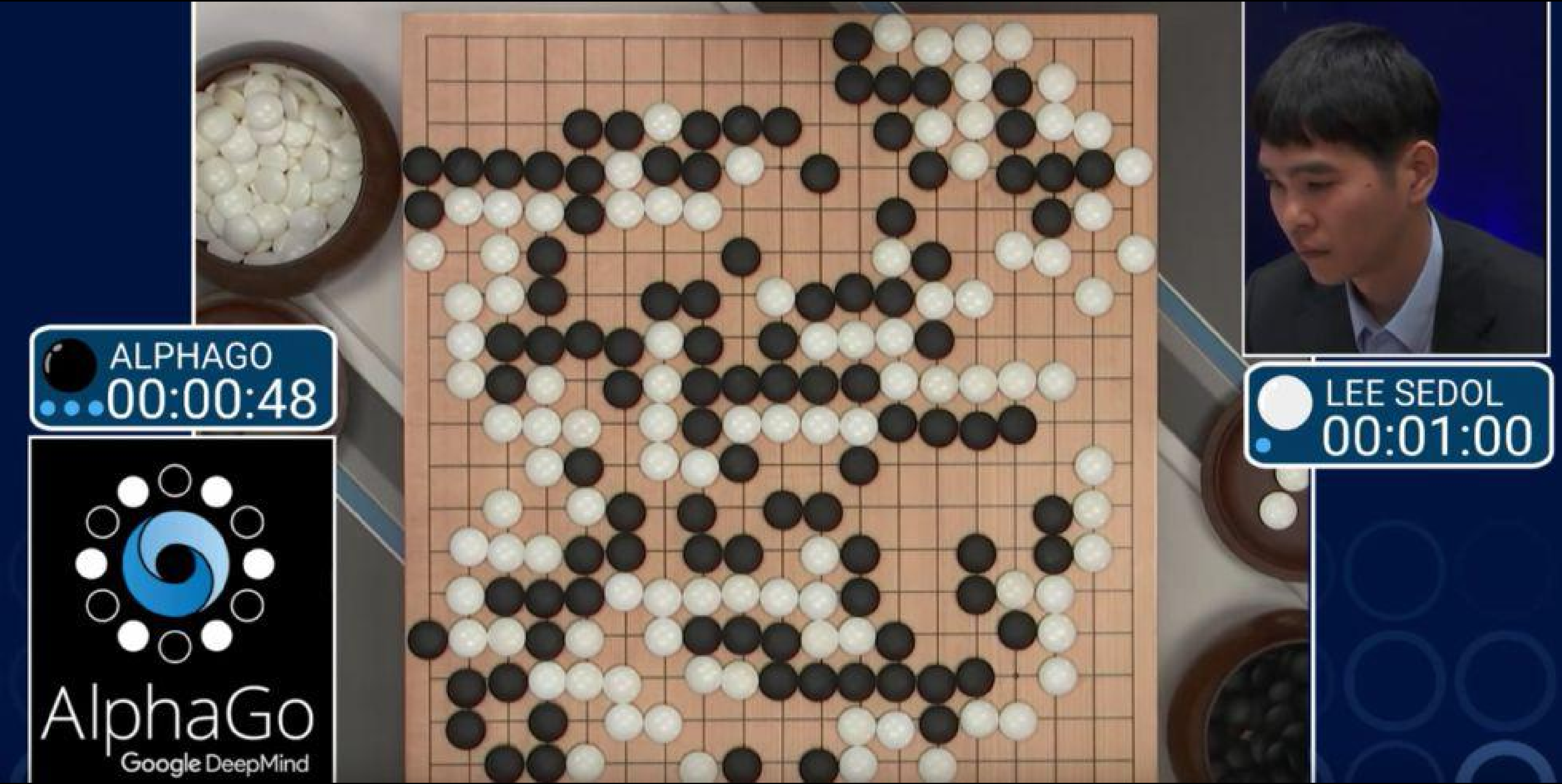
- Restrictive Assumptions
- Handles long horizons poorly
- Requires a simulator

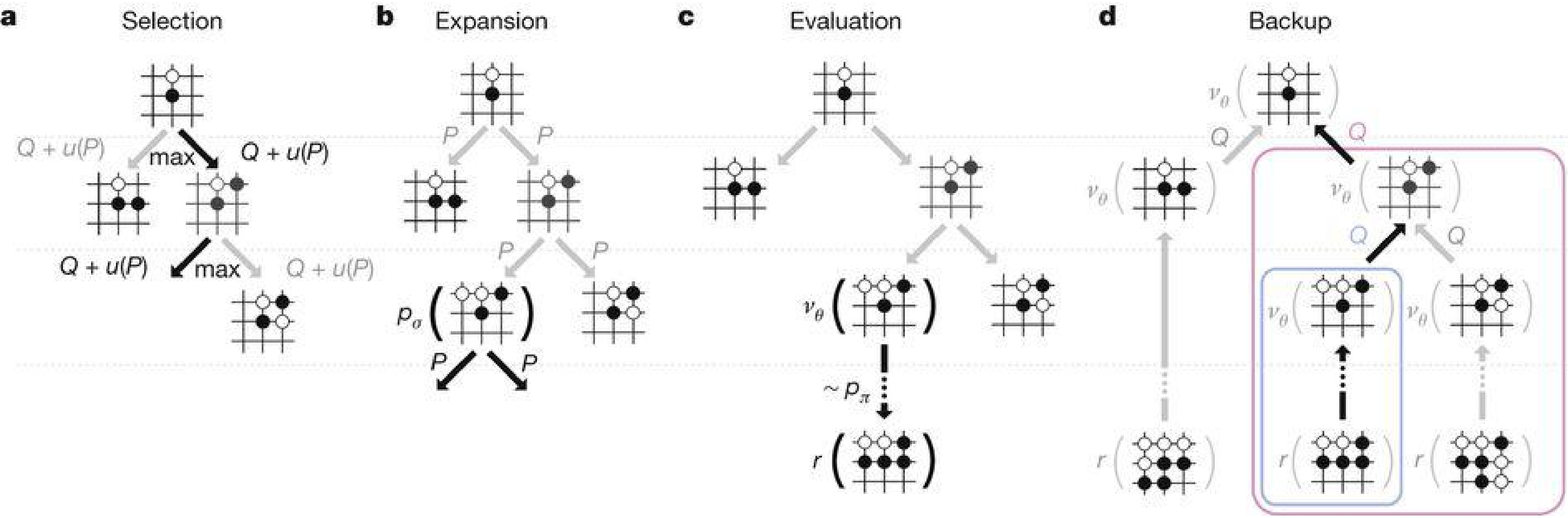
LSTM RNNs!



The problems with Q-learning

- Restrictive Assumptions
- Handles long horizons poorly
- Requires a simulator


ALPHAGO
00:00:48
LEE SEDOL
00:01:00
AlphaGo
Google DeepMind

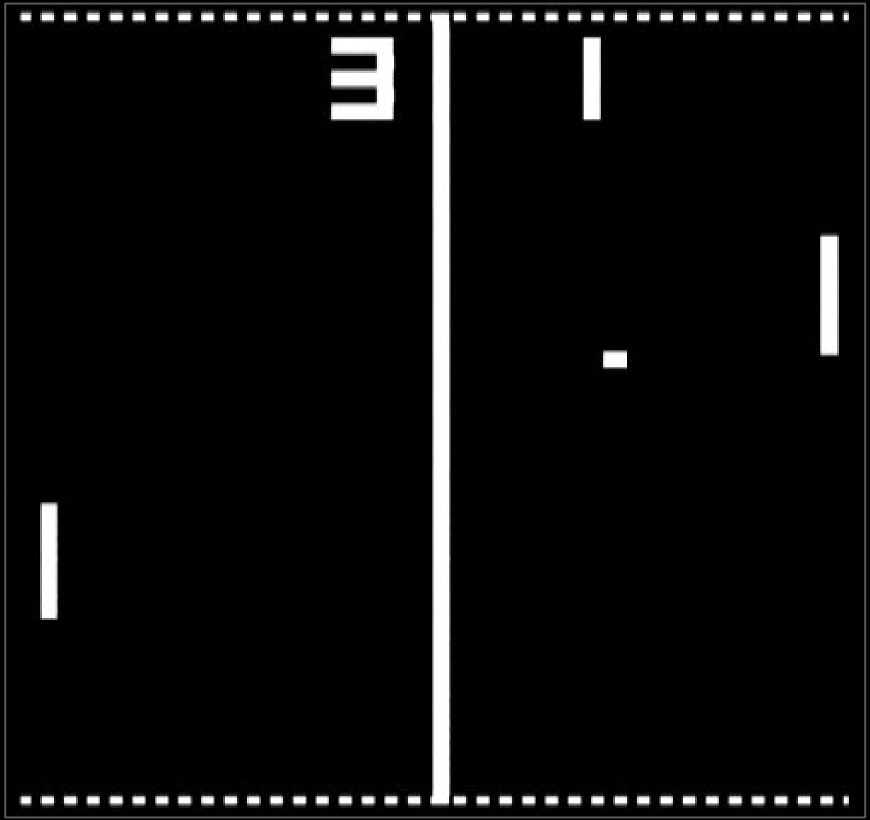

a
Selection
Q + u(P)
max
Q + u(P)
Q + u(P)
max
Q + u(P)
b
Expansion
P
P
P
P
p_σ
P
P
c
Evaluation
v_θ
~ p_π
r
d
Backup
v_θ
Q
Q
v_θ
v_θ
Q
Q
v_θ
v_θ
r
r
r

The problems with Q-learning

- Restrictive Assumptions
- Handles long horizons poorly
- Requires a simulator

Levine et al, 2016

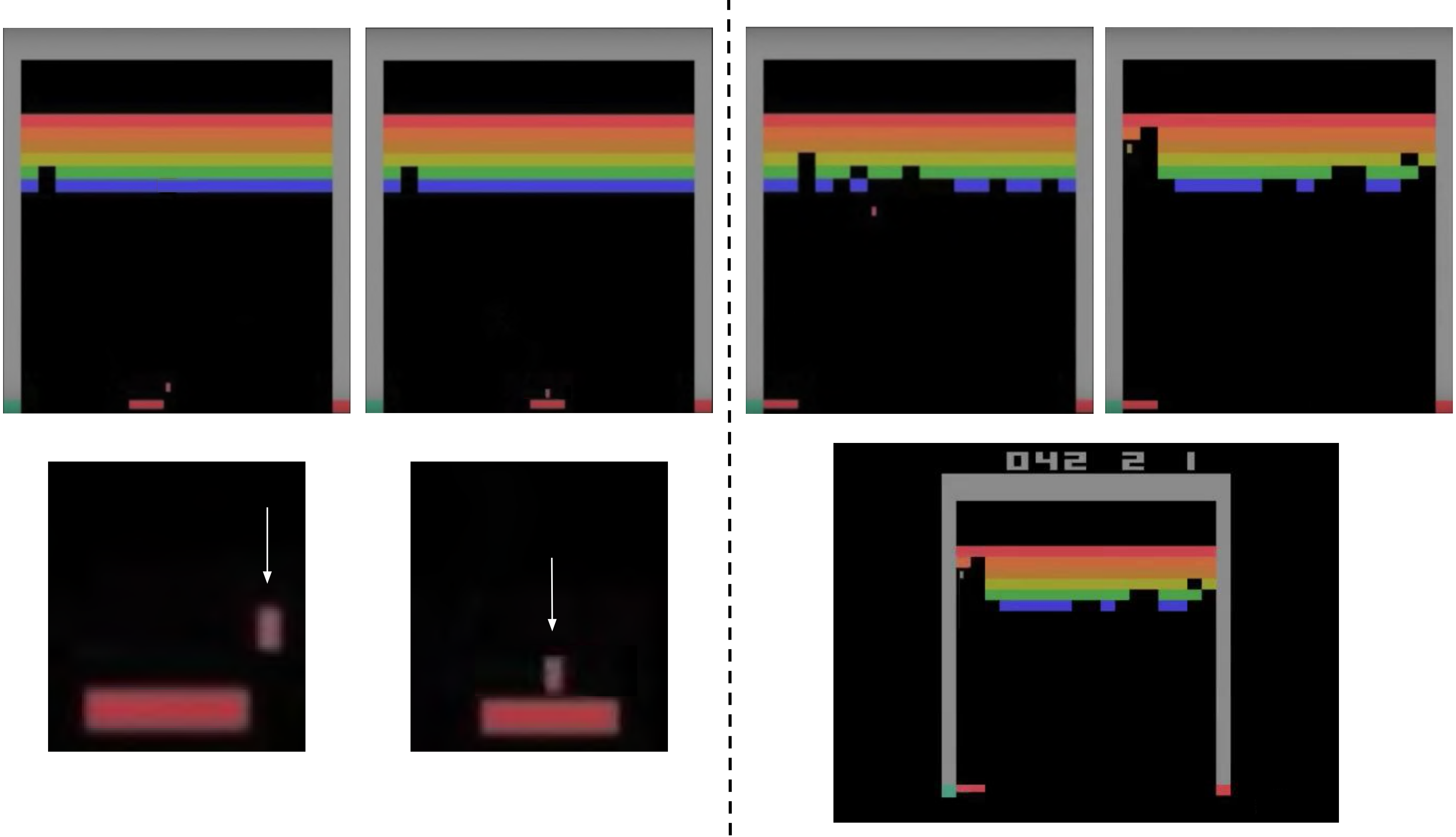

POLICE


3
1


B
2.13mi



10 #
Boston Dynamics


042 2 1

function ϵ-GREEDY-Q-LEARNING

  Initialize θ

  $\epsilon =$ some tiny number

  while not bored yet **do**

    $p =$ randf(0,1)

    if $p < \epsilon$ **then**

      Choose random action a and store results r, s_{new}

    else

      Choose $a = \arg\max_{a'} Q(s, a')$, and store results r, s_{new}

    Compute $\nabla_\theta E_Q = \nabla_\theta \left\| \widehat{Q}(s, a|\theta) - \left(r + \gamma \max_{a'} \widehat{Q}(s, a'|\theta_{old}) \right) \right\|$

    $\theta = \theta - \eta \nabla_\theta E_Q$

    $s = s_{new}$ (or s_0 if episode ended)

    $\theta_{old} = \theta$