very

Short tutorial on deep reinforcement learning

Brenden Petersen

LLNL-PRES-774209
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

What is deep reinforcement learning?

Given an

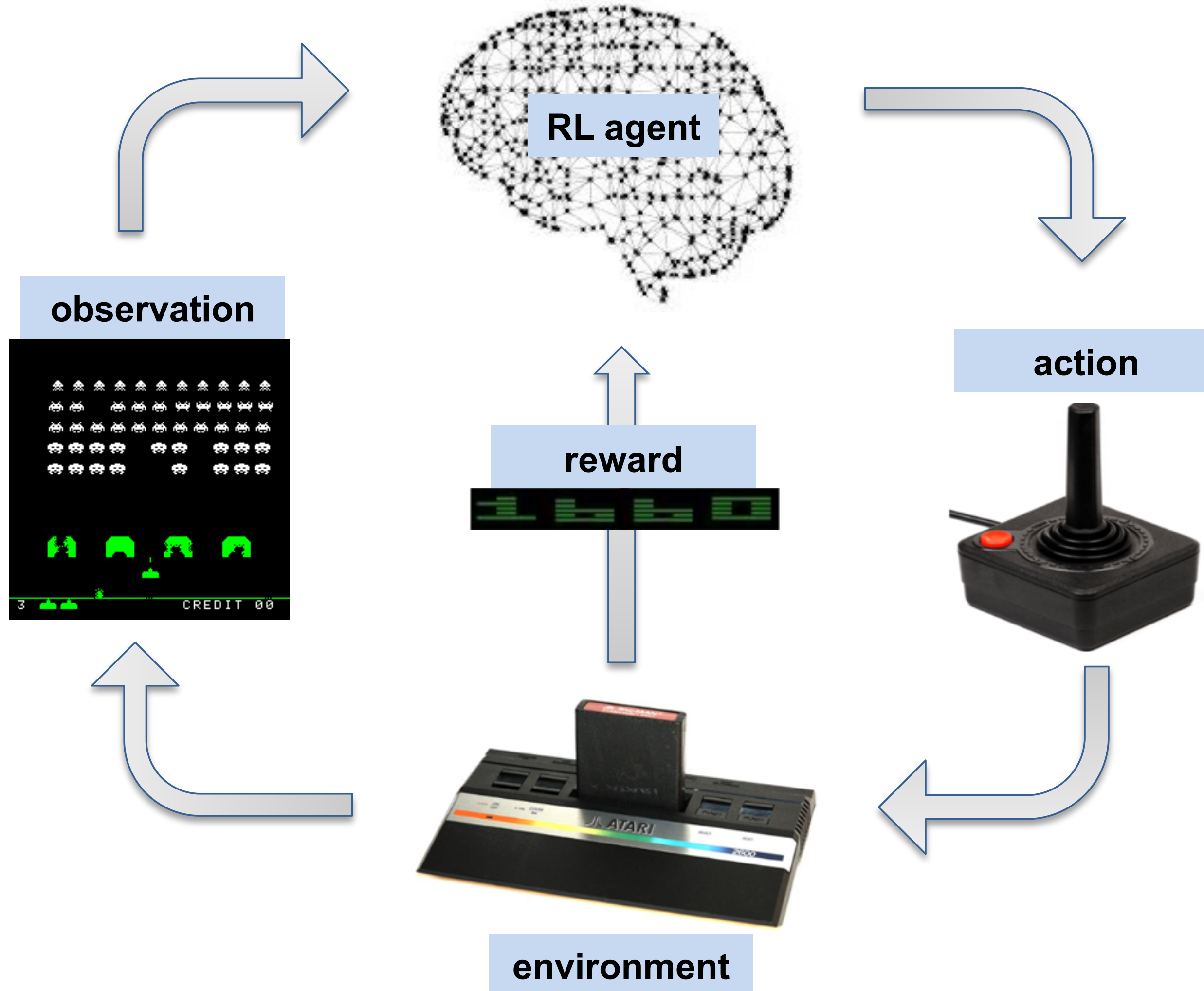
observation

choose the

action

expected to maximize the cumulative

reward



What can you do with DRL?

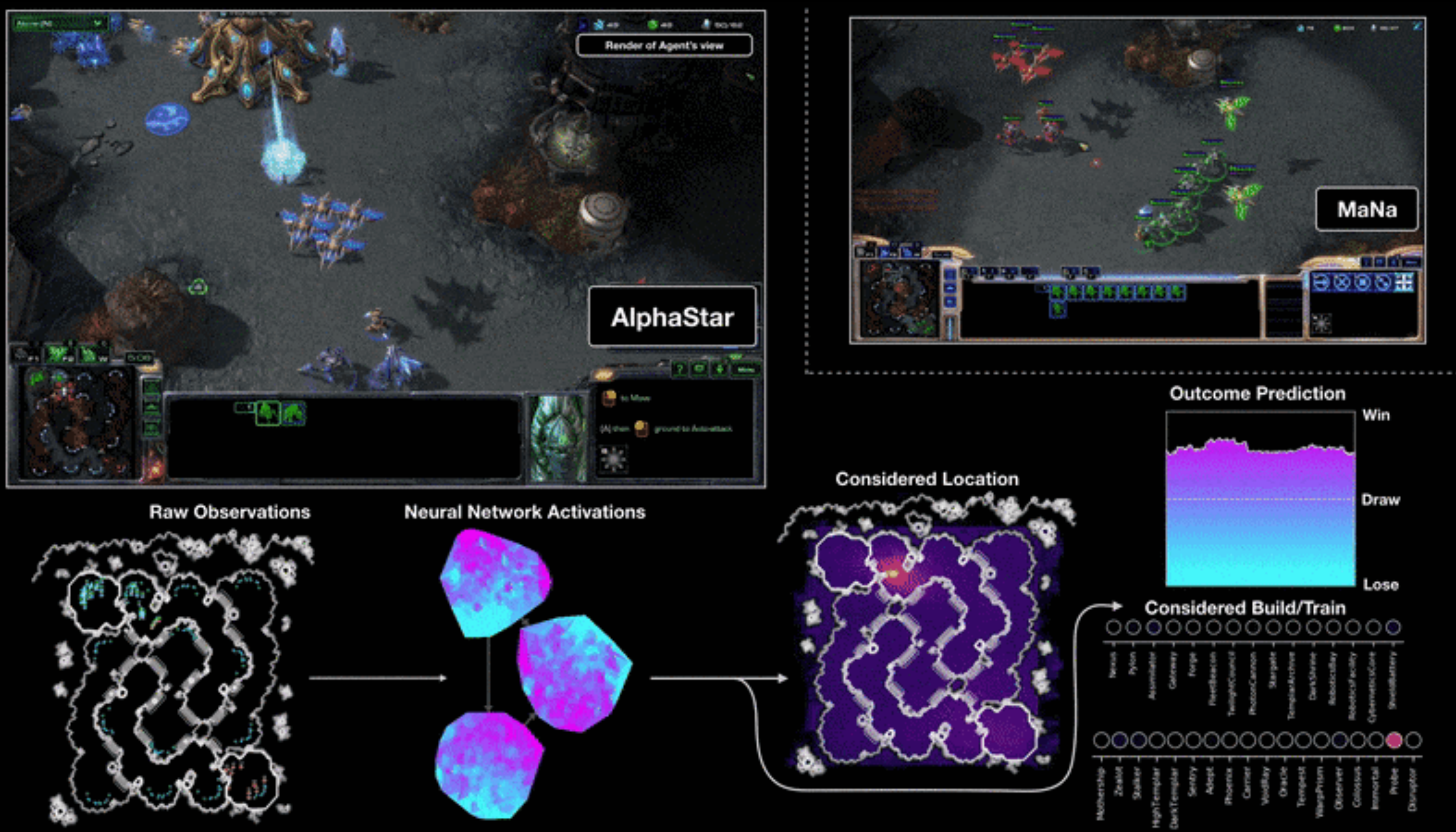
- Play video games better than a human

Observations
Pixels

Actions
{9 directions} x {Fire}

Reward
Game score


GET READY
STAGE 1
COLLECT 50 COOL POINTS!
HIGH SCORE
Day 1
TERRAPORT
5:00



Render of Agent's view
AlphaStar
MaNa
Outcome Prediction
Win
Draw
Lose
Considered Location
Raw Observations
Neural Network Activations
Considered Build/Train

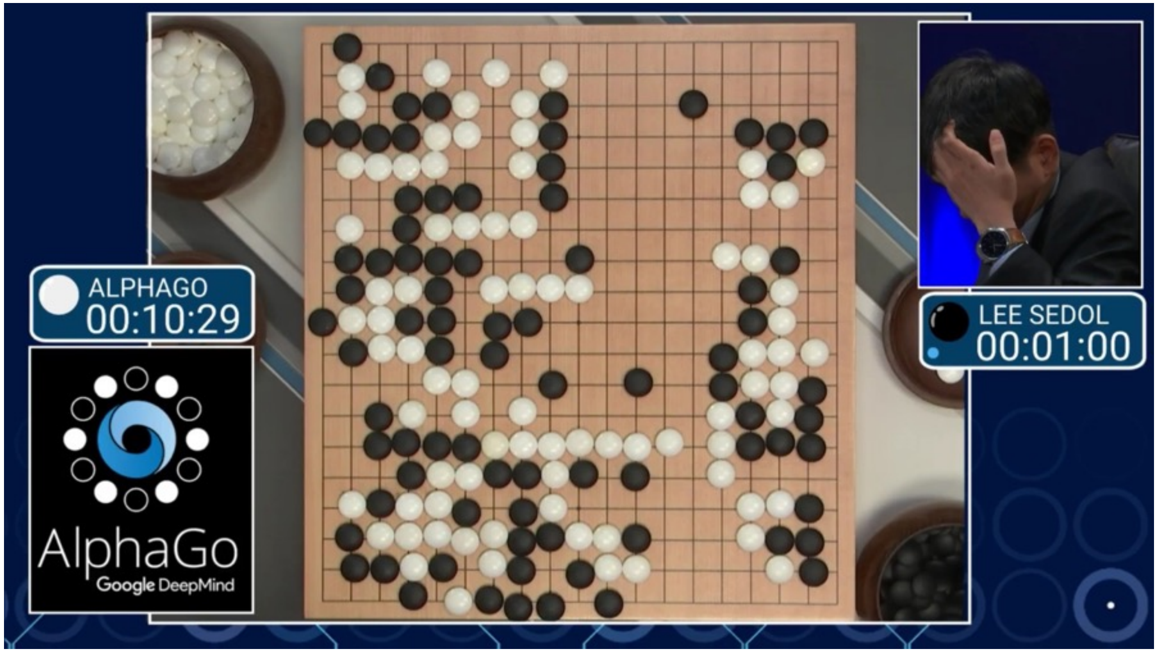
What can you do with DRL?

- Play board games better than a human

Observations
{19 x 19 grid} x
{B, W, Empty}

Actions
{19 x 19 grid}

Reward
1 (win)
0 (otherwise)



What can you do with DRL?

dance??

~~skip~~

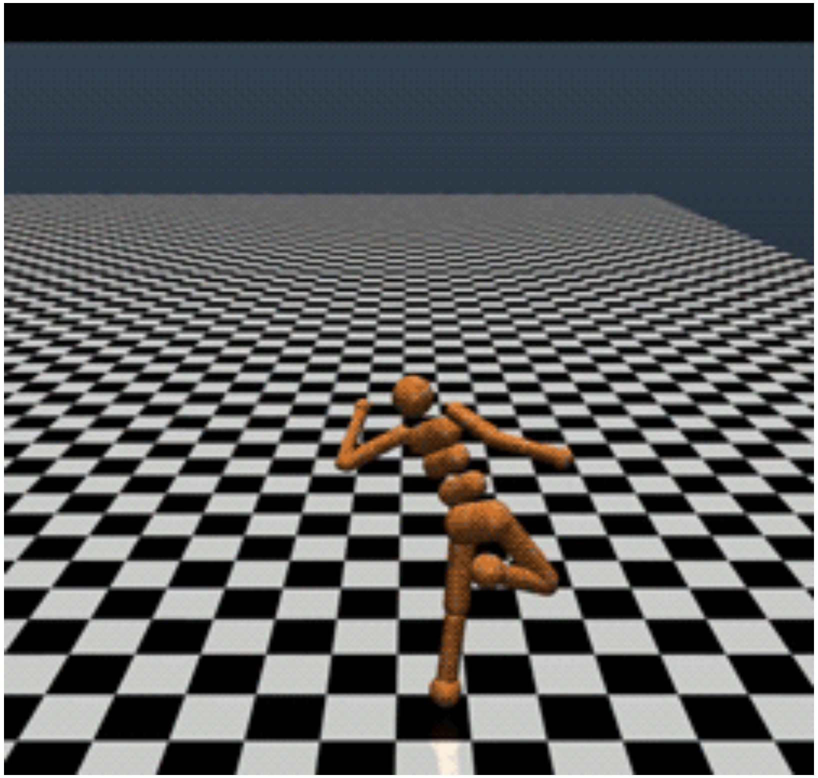
~~hop~~

- Teach a simulated robot how to ~~walk~~

Observations
Joint positions, angles, and velocities

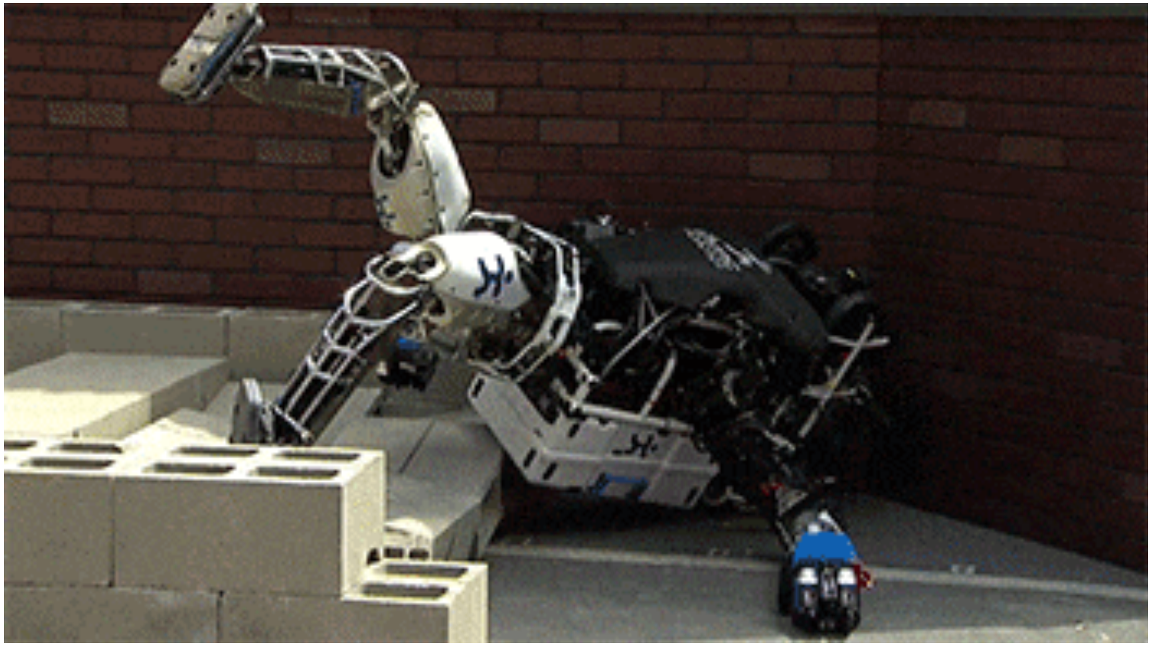
Actions
Joint torques

Reward
Distance travelled



What can you do with DRL?

- Teach a real robot how to walk
  - or perhaps not…



Formulating an RL problem

- An RL problem is formulated as a Markov decision process (MDP)

$$(\mathcal{S}, \mathcal{A}, \mathcal{E}, r, \gamma)$$

Observation space

Action space

State transition function or *environment*:
$\mathcal{E} = P_a(s, s') = \Pr(s'|s, a)$
This is often **unknown**.

Scalar reward signal
$r(s, a, s') \in \mathbb{R}$

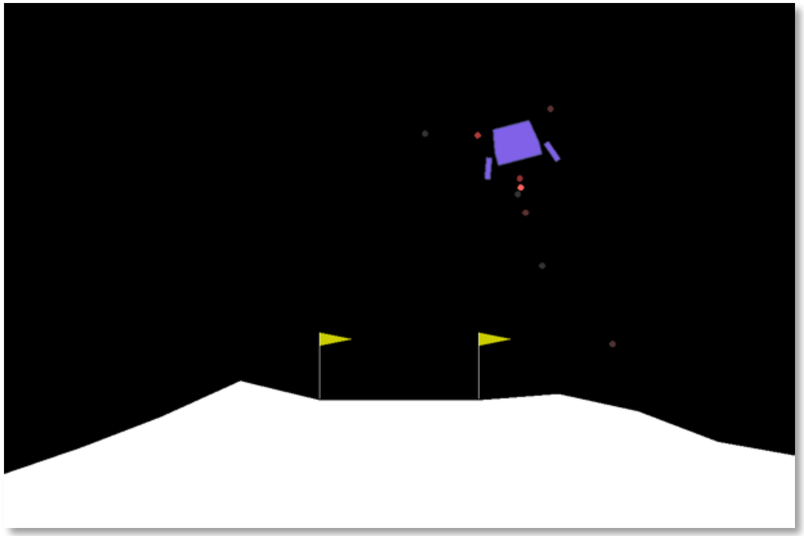
Discount factor
$\gamma \in (0,1]$

- MDPs satisfy the Markov property:

$$\Pr(s_{t+1}|s_0, a_0, s_1, a_1, \ldots, s_t, a_t) = \Pr(s_{t+1}|s_t, a_t)$$

Demo – LunarLander

- Goal: Get to the landing pad without crashing
- Observation space:
  - x, y coordinate
  - x, y velocity
  - Angle w.r.t. horizontal
  - Angular velocity
  - Left/right ground contact
- Action space:
  - Main thrusters (up): [0, 1]
  - Side thrusters (left/right): [-1, 1]
- Reward:
  - +100 for landing safely
  - –100 for crashing
  - Small penalty for fuel consumed
  - Bonus/penalty for moving closer to/further from goal



Key concepts & terminology

- The **return** is the cumulative discounted reward

$$R = \sum_{t=0}^{T} \gamma^t r_t$$

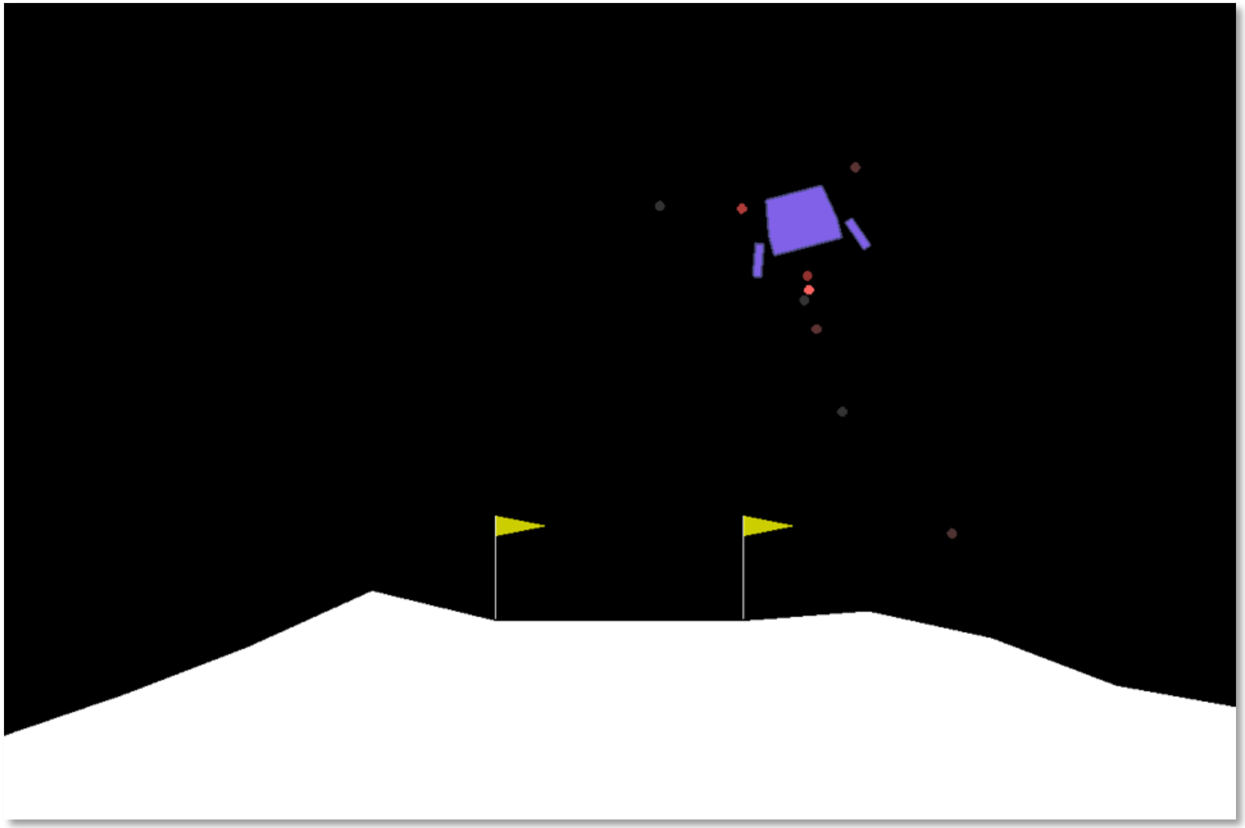
- A **policy** defines a mapping from state to action

$$\pi : \mathcal{S} \mapsto \mathcal{A}$$

- A **value function** estimates “how good it is” to be in a certain state

$$V_\pi(s) = \mathbb{E}\left[R | s, \pi\right]$$
$$Q_\pi(s, a) = \mathbb{E}\left[R | s, a, \pi\right]$$

How’s our LunarLander doing?



How do you learn an optimal policy?

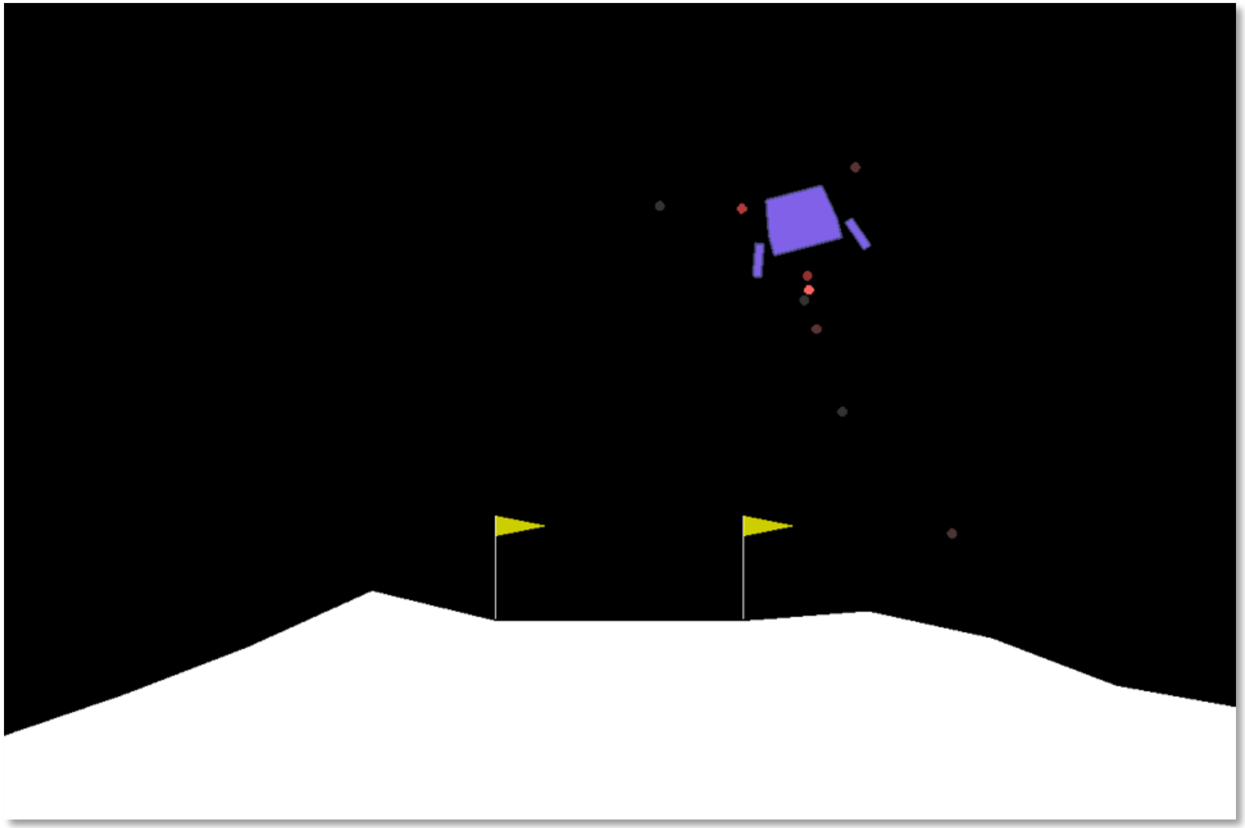
- If you can estimate the *Q* value, the optimal policy is to select the action with the largest *Q* value

$$\pi^\star(s) = \arg\max_a Q(s,a)$$

- We can **learn** a value function by iteratively updating our current estimate based on the reward we received

$$\Delta Q \sim \underbrace{Q(s,a)}_{\text{what you thought you'd get}} - \underbrace{\left(r + \gamma \max_a Q(s',a)\right)}_{\text{what you got}}$$

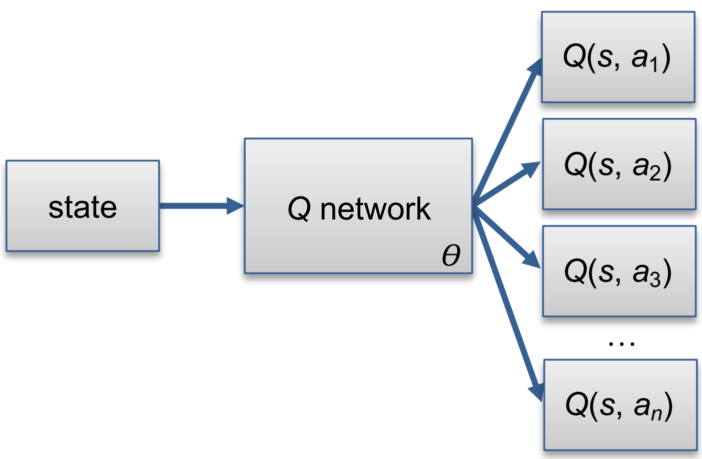
Are we landing yet??



From tabular to deep RL

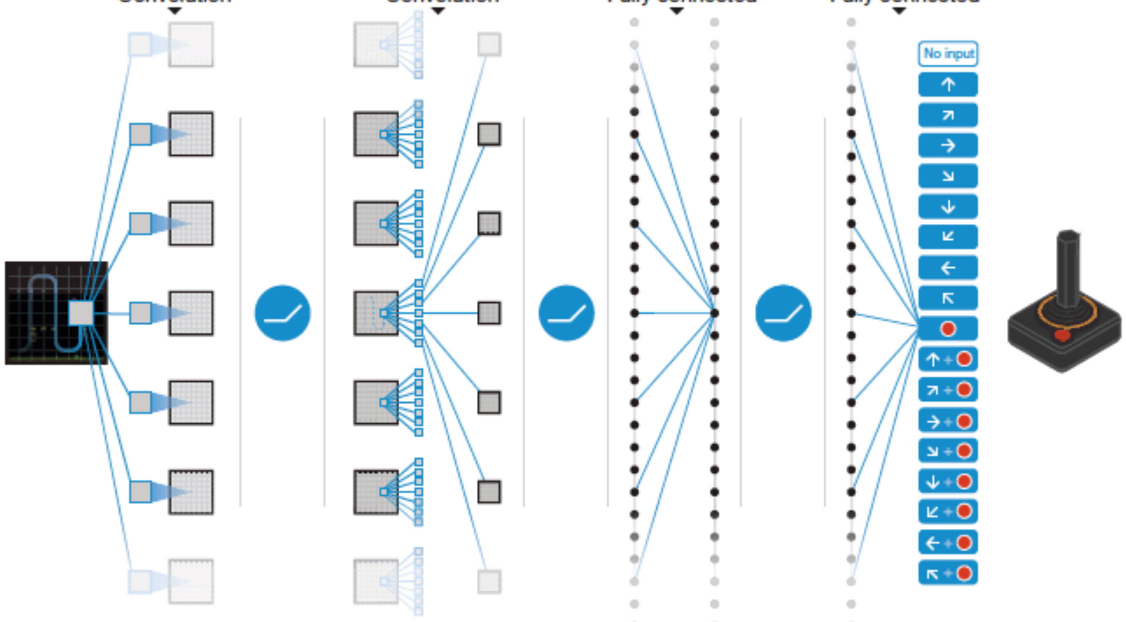
- Tabular (traditional) RL:
  - Learn the Q value separately for every state-action combination
  - Requires **discrete** states
  - Requires **discrete** actions
- Deep RL:
  - Learn a *function* that maps states to Q values
  - Works for **continuous** states
  - Requires **discrete** actions

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	…	a_m
s_1	1.21	2.23	1.42	2.42	5.32	3.14	6.37		1.30
s_2	4.56	3.33	1.43	3.16	2.67	2.53	2.22		3.16
s_3	4.46	5.66	2.16	3.45	2.53	6.35	2.64		3.56
s_4	4.77	3.16	7.77	3.54	9.01	6.46	3.26		4.26
…								⋱	
s_n	8.11	2.22	8.00	7.64	7.66	5.66	5.44		9.03



Playing Atari with DRL: Problem setup

- States = sequence of 4 images
- 18 possible actions
- *Q* function network architecture:



Space Invaders

Breakout

Frostbite

Playing Atari with DRL: Algorithm

Algorithm 1 Deep Q Networks (DQN)

Initialize replay buffer $\mathcal{D}$
Initialize Q network with random weights
for episode $= 1, M$ **do**
    for $t = 1, T$ **do**
        With probability ϵ select random action...
        ...otherwise select $a = \max_a Q(s, a; \theta)$
        Execute action a in environment; receive s', r
        Store transition (s, a, s', r) in $\mathcal{D}$
        Sample minibatch from $\mathcal{D}$
        Perform gradient step: $\nabla_\theta \frac{1}{2} \left[Q(s, a; \theta) - (r + \gamma \max_a Q(s', a; \theta))\right]^2$

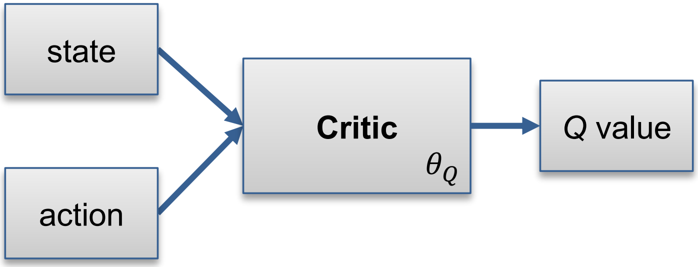
Tabular:

$$\Delta Q \sim Q(s, a) - \Big(r + \gamma \max_a Q(s', a)\Big)$$

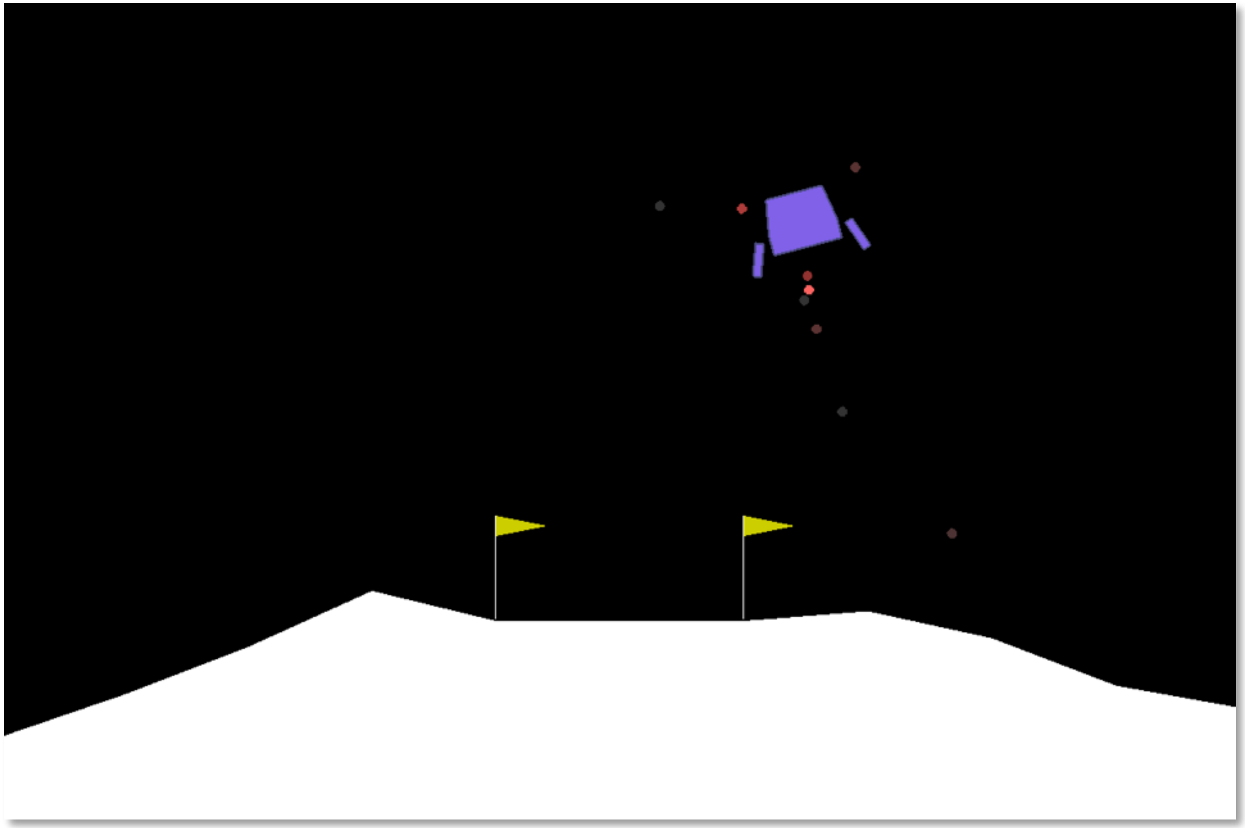
What about continuous action spaces?

- One network selects an action
  - Policy network or “actor”
  - Updated using the *policy gradient theorem*

- Another network evaluates the action
  - *Q* network or “critic”
  - Updated the same way as DQN

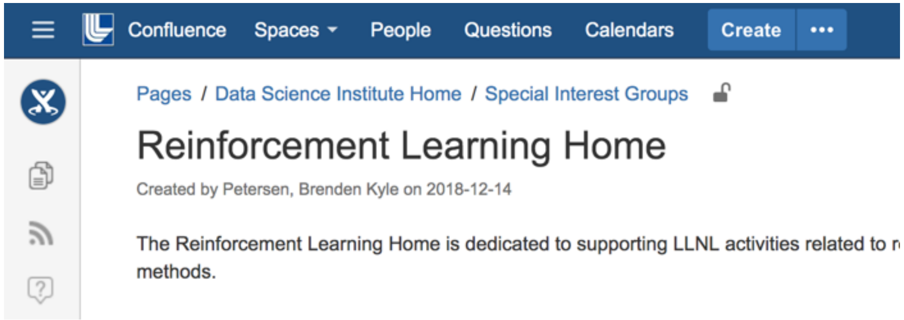


Have we converged?



Want to learn more?

- LLNL Reinforcement Learning reading group
  - Wednesdays 2 – 3
  - B170 or B155



- OpenAI Spinning Up
  - Web-based "fast-track to DRL" tutorial
  - spinningup.openai.com

- Sutton & Barto
  - Free online PDF of textbook