# DL + RL =
# Deep Reinforcement Learning

(Slides by Svetlana Lazebnik, B Ravindran, David Silver)

# Function approximation

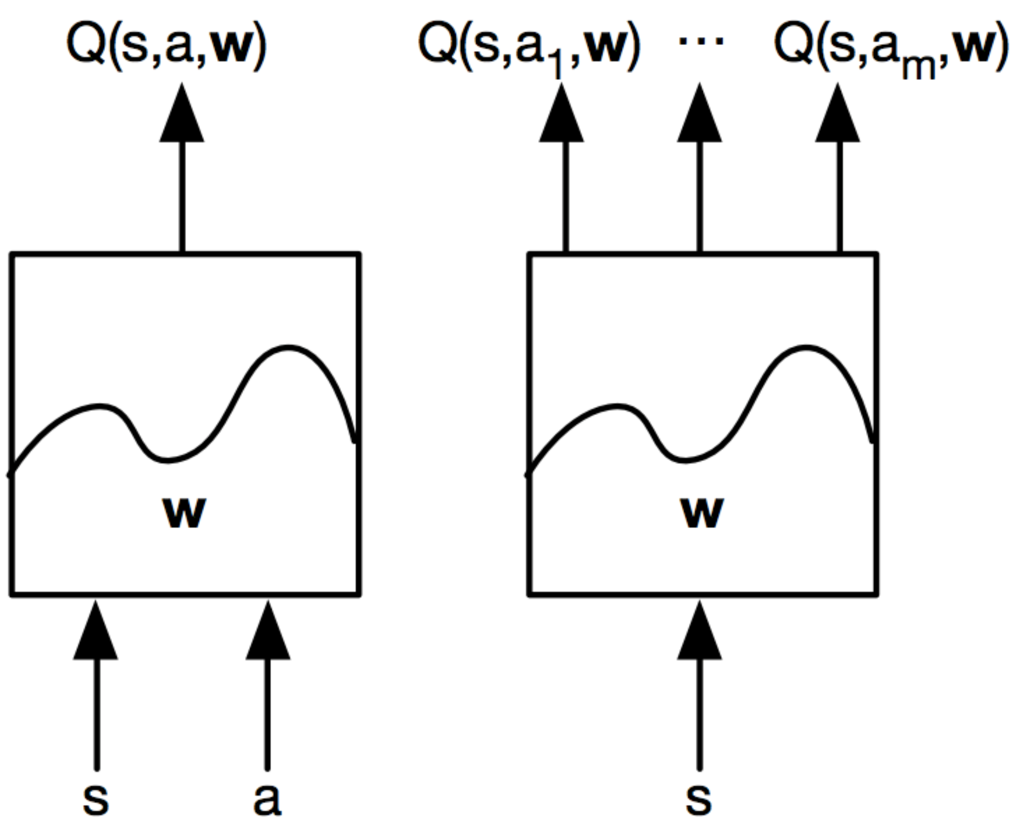
- So far, we've assumed a *lookup table* representation for utility function U(s) or action-utility function Q(s,a)
- This does not work if the state space is really large or continuous
- Alternative idea: approximate the utilities or Q values using parametric functions and automatically learn the parameters:

$$V(s) \approx \hat{V}(s; w)$$

$$Q(s,a) \approx \hat{Q}(s,a; w)$$

# Deep Q learning

- Train a deep neural network to output Q values:




Source: [D. Silver]

# Deep Q learning

- Regular TD update: "nudge" Q(s,a) towards the target

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(\boxed{R(s) + \gamma \max_{a'} Q(s',a')} - Q(s,a)\right)$$

- Deep Q learning: encourage estimate to match the target by minimizing squared error:

$$L(w) = \left(\underbrace{\boxed{R(s) + \gamma \max_{a'} Q(s',a';w)}}_{\text{target}} - \underbrace{\boxed{Q(s,a;w)}}_{\text{estimate}}\right)^2$$

# Deep Q learning

- Regular TD update: "nudge" Q(s,a) towards the target

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(\boxed{R(s) + \gamma \max_{a'} Q(s',a')} - Q(s,a)\right)$$

- Deep Q learning: encourage estimate to match the target by minimizing squared error:

$$L(w) = \Big(\underbrace{\boxed{R(s) + \gamma \max_{a'} Q(s',a';w)}}_{\text{target}} - \underbrace{\boxed{Q(s,a;w)}}_{\text{estimate}}\Big)^2$$

- Compare to supervised learning:

$$L(w) = \big(y - f(x;w)\big)^2$$

  – Key difference: the target in Q learning is also moving!

# Online Q learning algorithm

- Observe experience (s,a,s', r)
- Compute target $y = r + \gamma \max_{a'} Q(s',a';w)$
- Update weights to reduce the error

$$L = \left(y - Q(s,a;w)\right)^2$$

- Gradient: $\nabla_w L = \left(Q(s,a;w) - y\right) \nabla_w Q(s,a;w)$

- Weight update: $w \leftarrow w - \alpha \nabla_w L$

- This is called *stochastic gradient descent* (SGD)

# Dealing with training instability

- Challenges
  - Target values are not fixed
  - Successive experiences are correlated and dependent on the policy
  - Policy may change rapidly with slight changes to parameters, leading to drastic change in data distribution
- Solutions
  - Freeze target Q network
  - Use *experience replay*

Mnih et al. Human-level control through deep reinforcement learning, *Nature* 2015

# Experience replay

- At each time step:
  - Take action $a_t$ according to epsilon-greedy policy
  - Store experience ($s_t$, $a_t$, $r_{t+1}$, $s_{t+1}$) in *replay memory buffer*
  - Randomly sample *mini-batch* of experiences from the buffer

| |
|---|
| $s_1, a_1, r_2, s_2$ |
| $s_2, a_2, r_3, s_3$ |
| $s_3, a_3, r_4, s_4$ |
| $\ldots$ |
| $s_t, a_t, r_{t+1}, s_{t+1}$ |

$s_t, a_t, r_{t+1}, s_{t+1} \quad \rightarrow$

Mnih et al. Human-level control through deep reinforcement learning, *Nature* 2015

# Experience replay

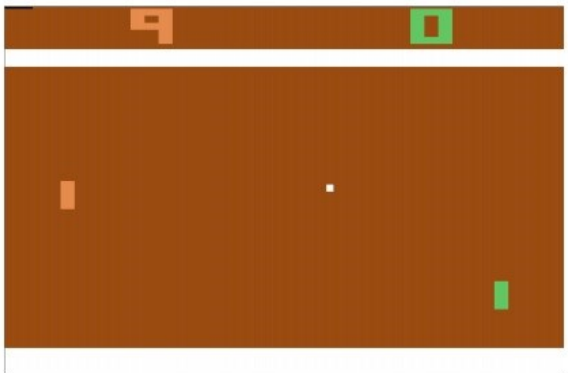
- At each time step:
  - Take action $a_t$ according to epsilon-greedy policy
  - Store experience ($s_t$, $a_t$, $r_{t+1}$, $s_{t+1}$) in *replay memory buffer*
  - Randomly sample *mini-batch* of experiences from the buffer
  - Perform update to reduce objective function

$$\mathbf{E}_{s,a,s'}\left[\left(R(s)+\gamma \max_{a'} \boxed{Q(s',a';w^{-})} - Q(s,a;w)\right)^2\right]$$

Keep parameters of *target network* fixed, update every once in a while

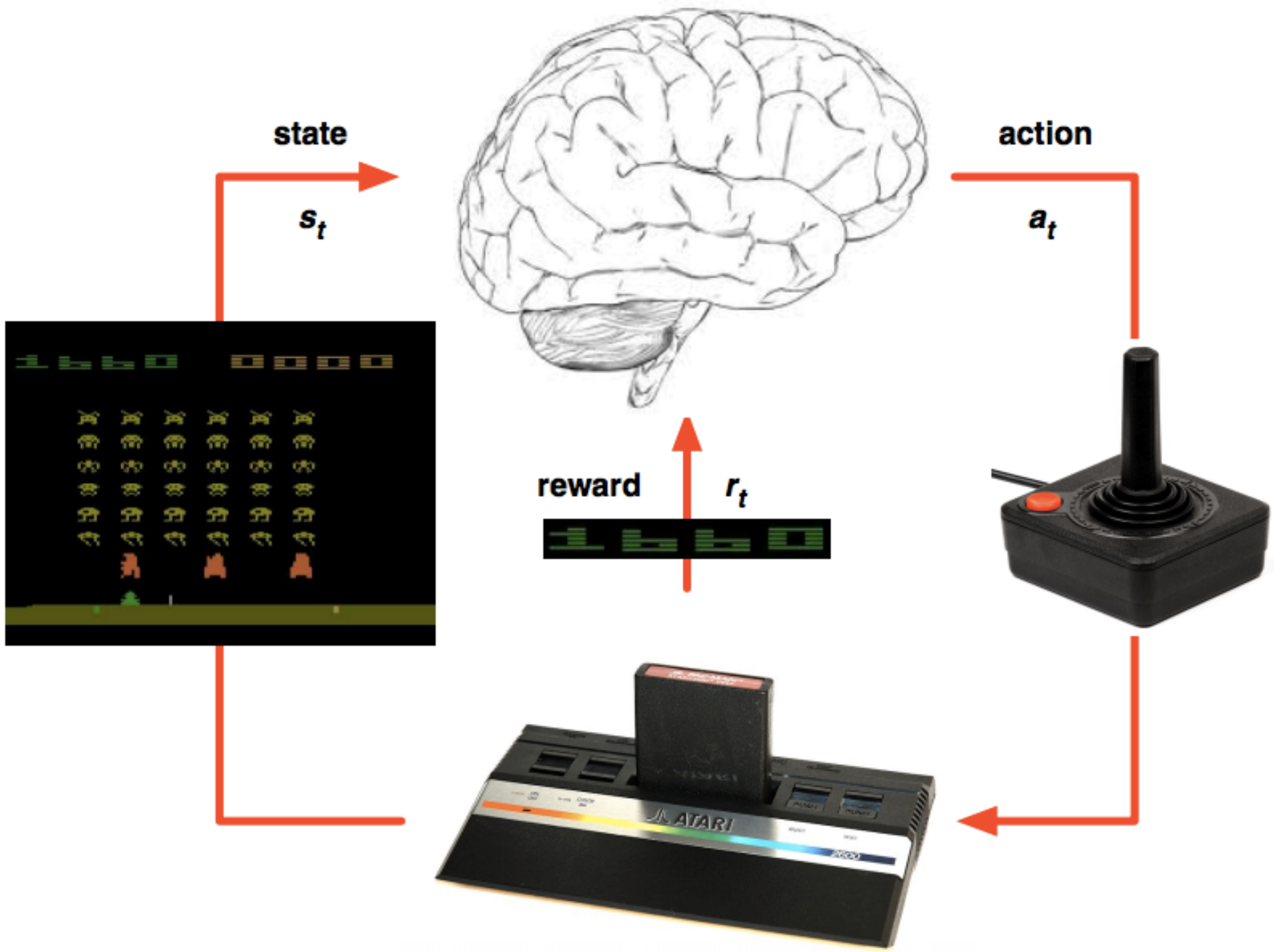
Mnih et al. Human-level control through deep reinforcement learning, *Nature* 2015

# Atari



- Learnt to play from video input
  - from scratch
- Used a complex *neural network!*
  - Considered one of the hardest learning problems solved by a computer.
- More importantly *reproducible!!*

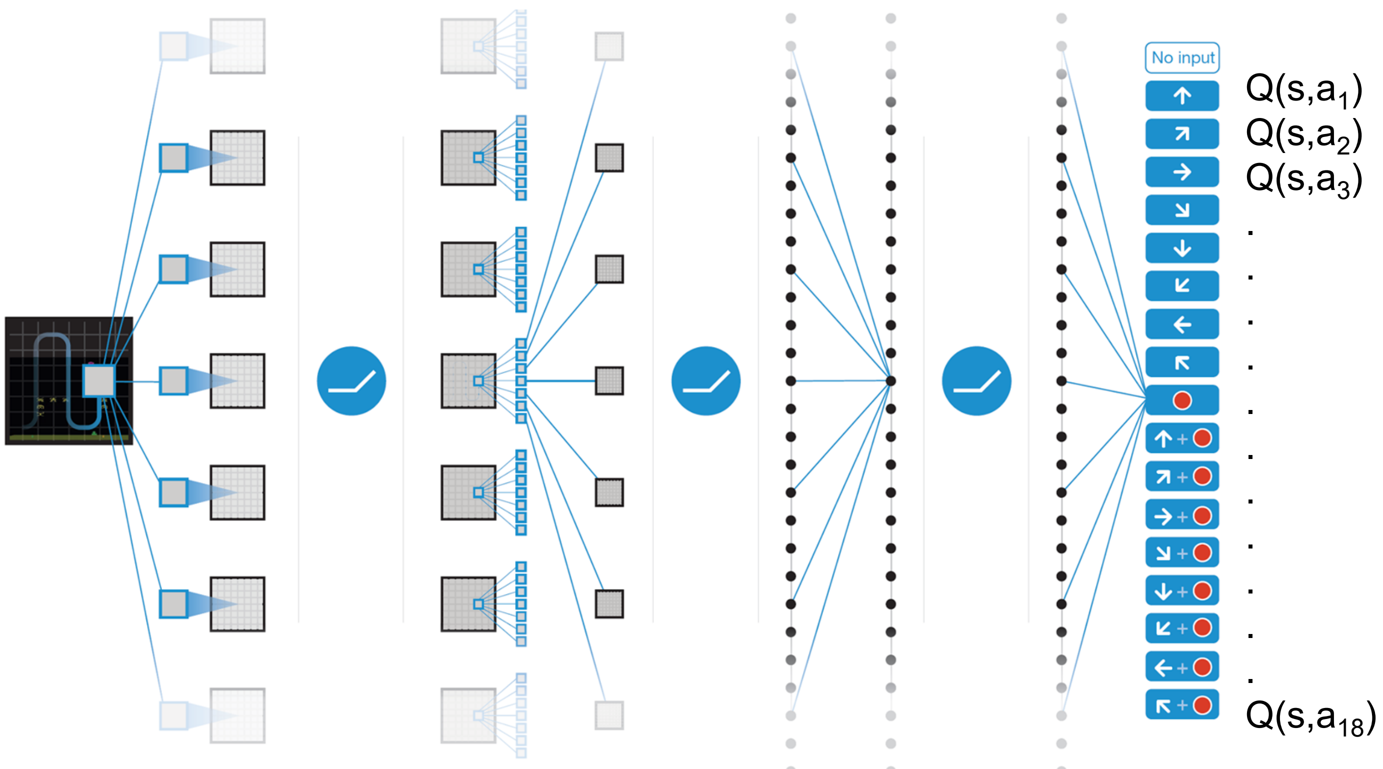
# Deep Q learning in Atari




Mnih et al. Human-level control through deep reinforcement learning, *Nature* 2015

# Deep Q learning in Atari

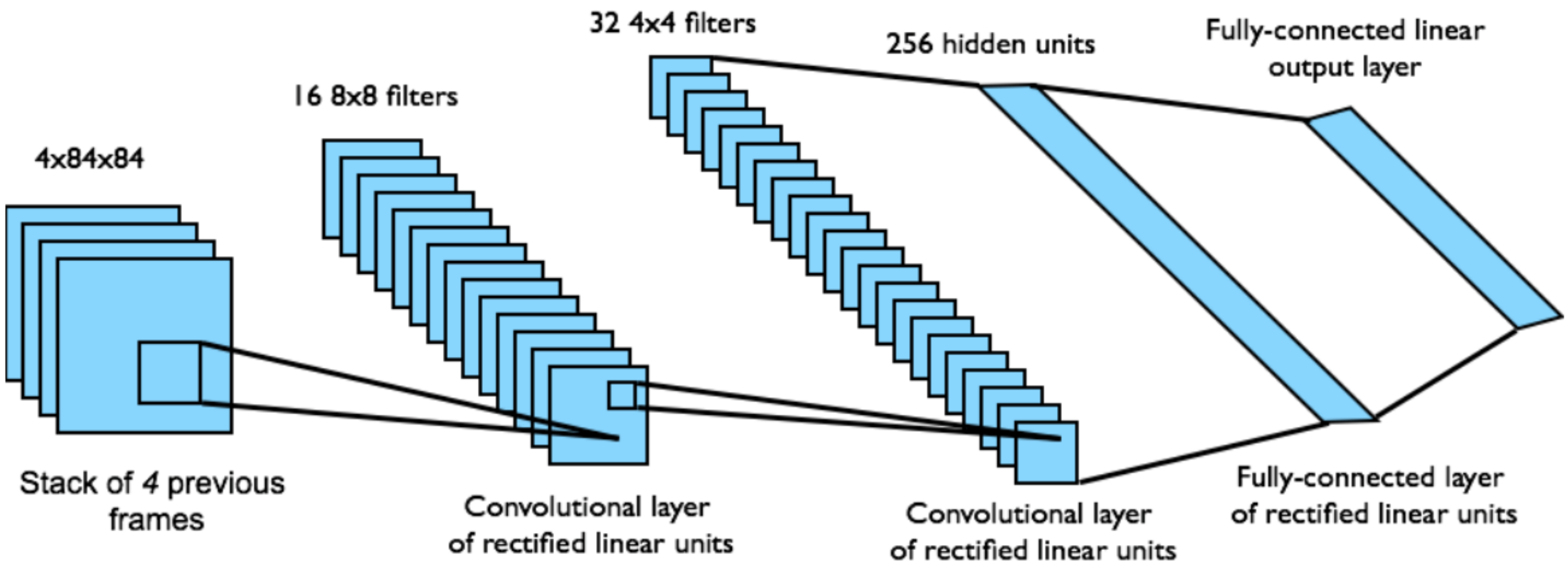
- End-to-end learning of Q(s,a) from pixels s
- Output is Q(s,a) for 18 joystick/button configurations
- Reward is change in score for that step




Mnih et al. Human-level control through deep reinforcement learning, *Nature* 2015

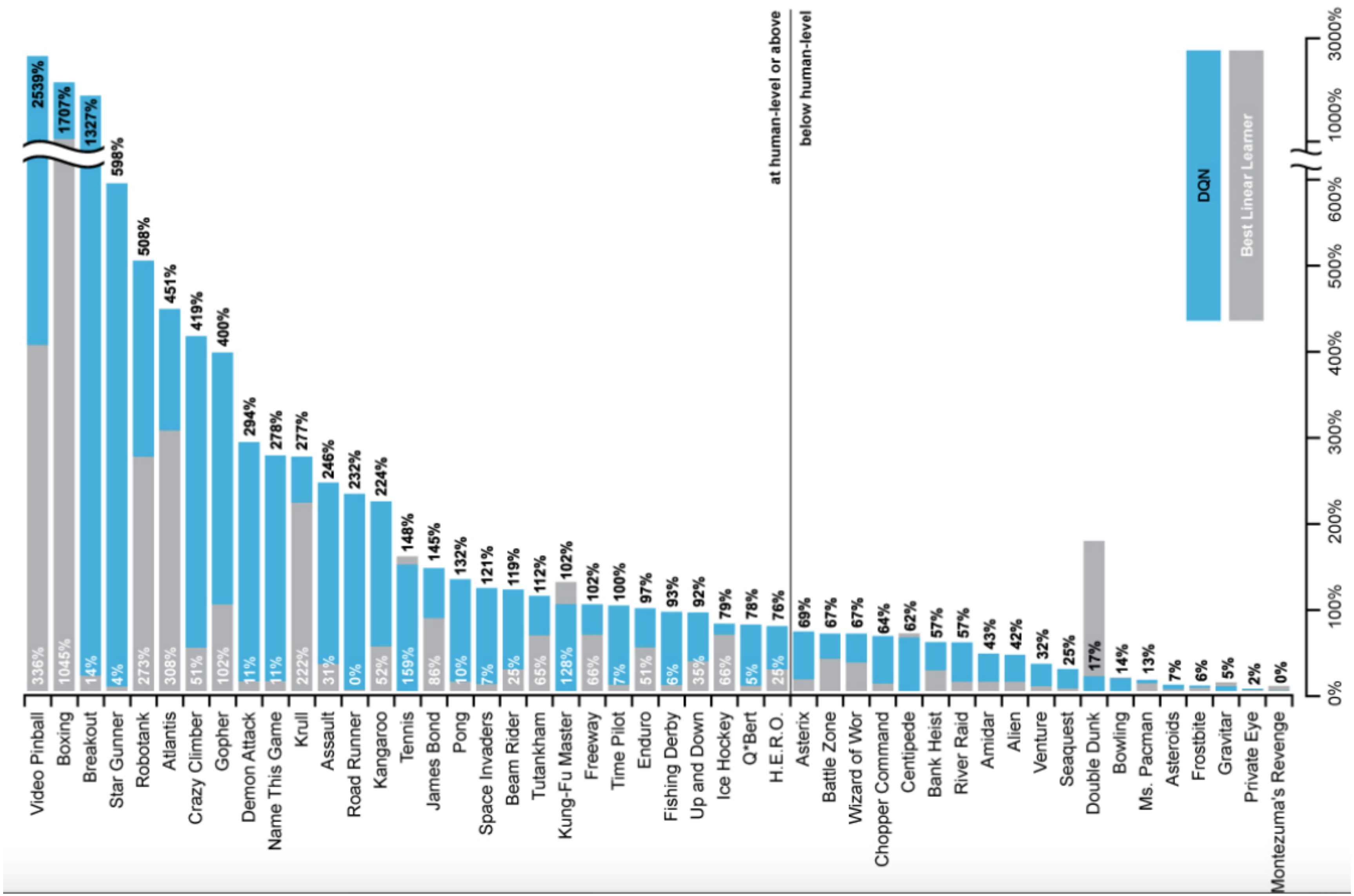
# Deep Q learning in Atari

- Input state s is stack of raw pixels from last 4 frames
- Network architecture and hyperparameters fixed for all games




Mnih et al. Human-level control through deep reinforcement learning, *Nature* 2015

# Deep Q learning in Atari

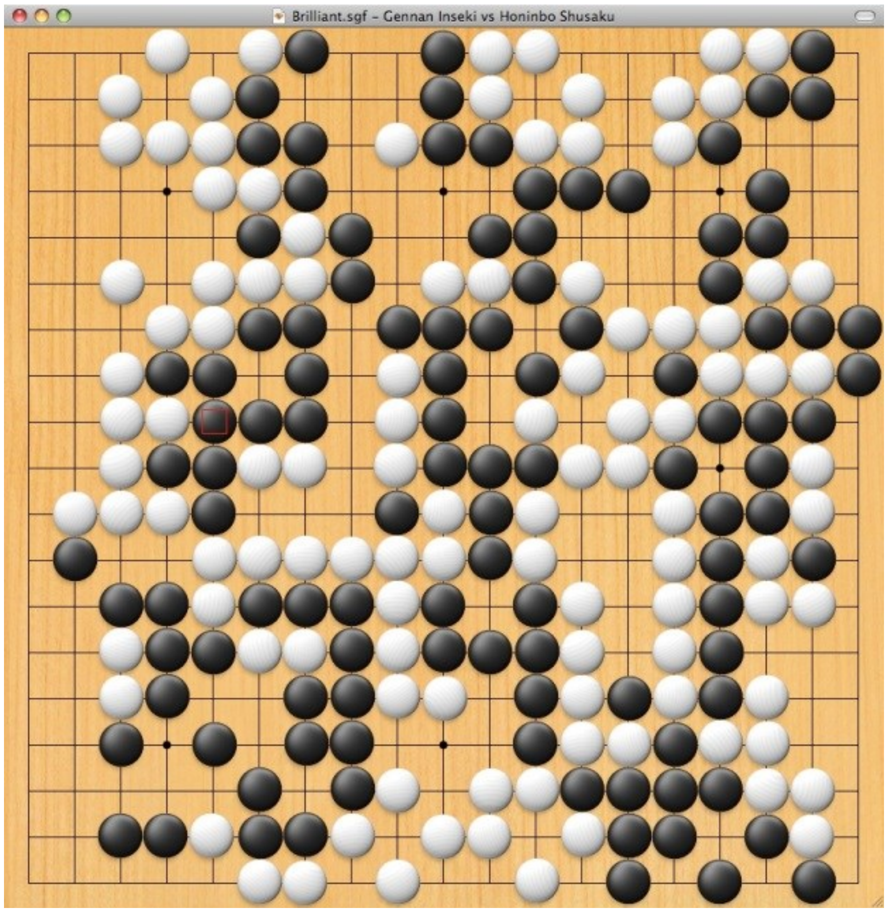



Mnih et al. Human-level control through deep reinforcement learning, *Nature* 2015

# Breakout demo

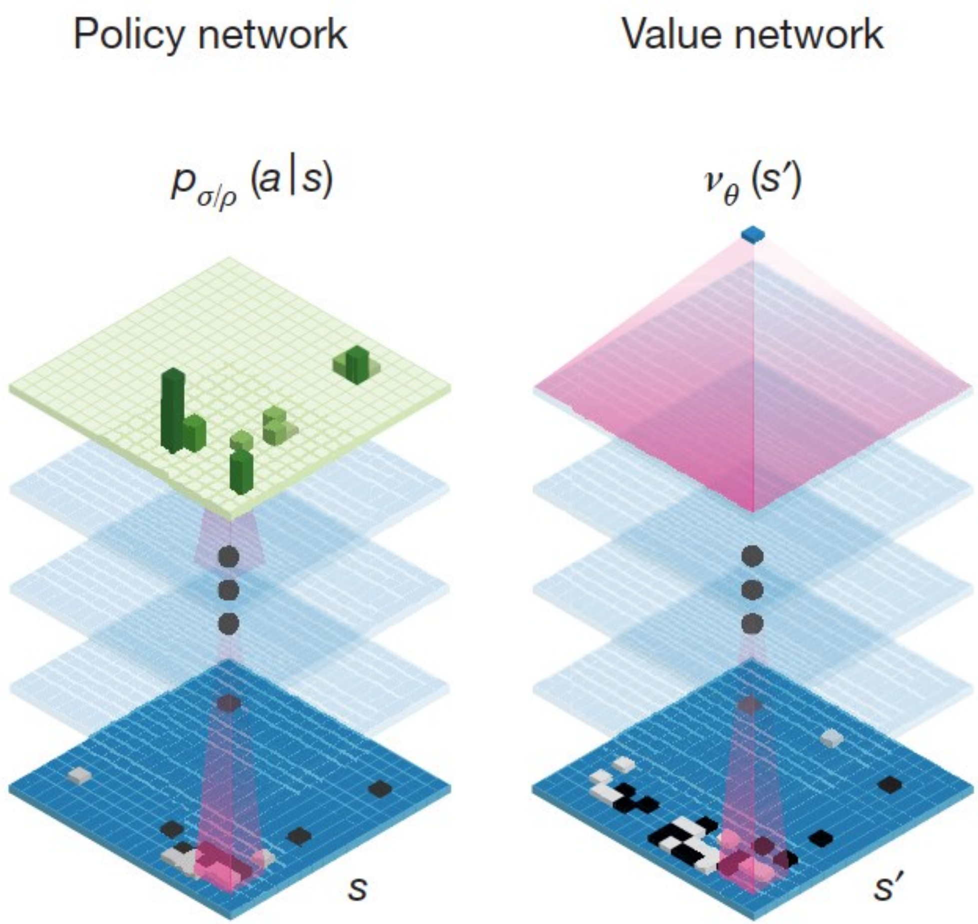

https://www.youtube.com/watch?v=TmPfTpjtdgg

# Playing Go



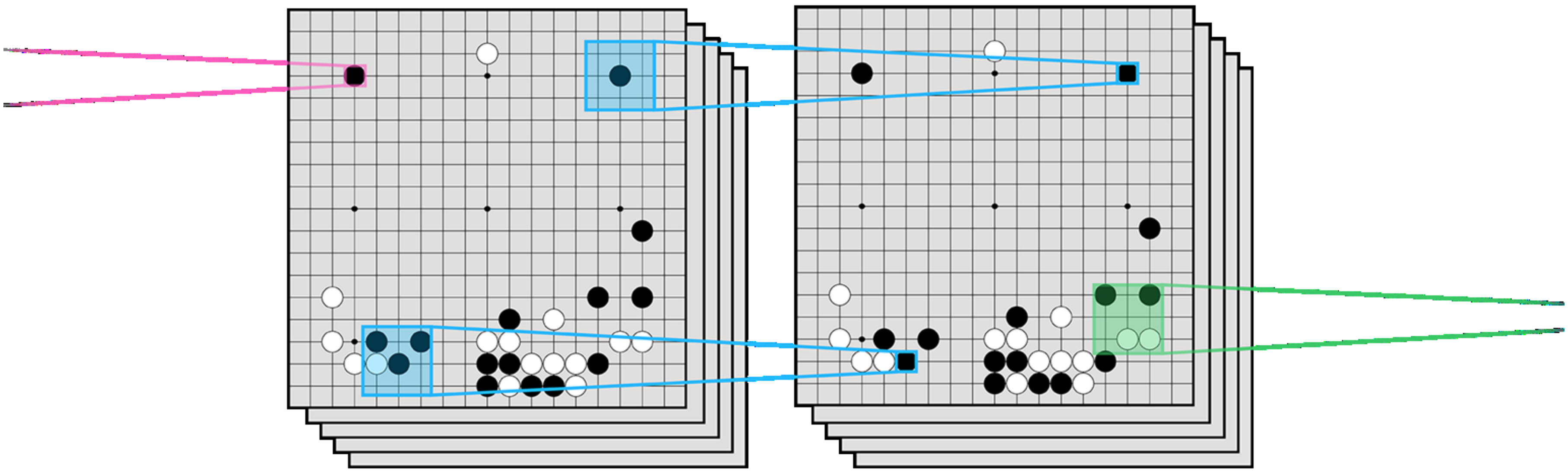

- Go is a known (and deterministic) environment
- Therefore, learning to play Go involves solving a known MDP
- Key challenges: huge state and action space, long sequences, sparse rewards

# Review: AlphaGo




- **Policy network:** initialized by supervised training on large amount of human games
- **Value network:** trained to predict outcome of game based on self-play
- Networks are used to guide Monte Carlo tree search (MCTS)

D. Silver et al., Mastering the Game of Go with Deep Neural Networks and Tree Search, Nature 529, January 2016



# Summary

- Deep Learning Strengths
  - universal approximators: learn non-trivial fns
  - compositional models ~similar to human brain
  - universal representation across modalities
  - discover features automagically
    - in a task-specific manner
    - features not limited by human creativity
- Deep Learning Weaknesses
  - resource hungry (data/compute)
  - Uninterpretable
- Deep RL: replace value/policy tables by deep nets
  - Great success in Go, Atari.