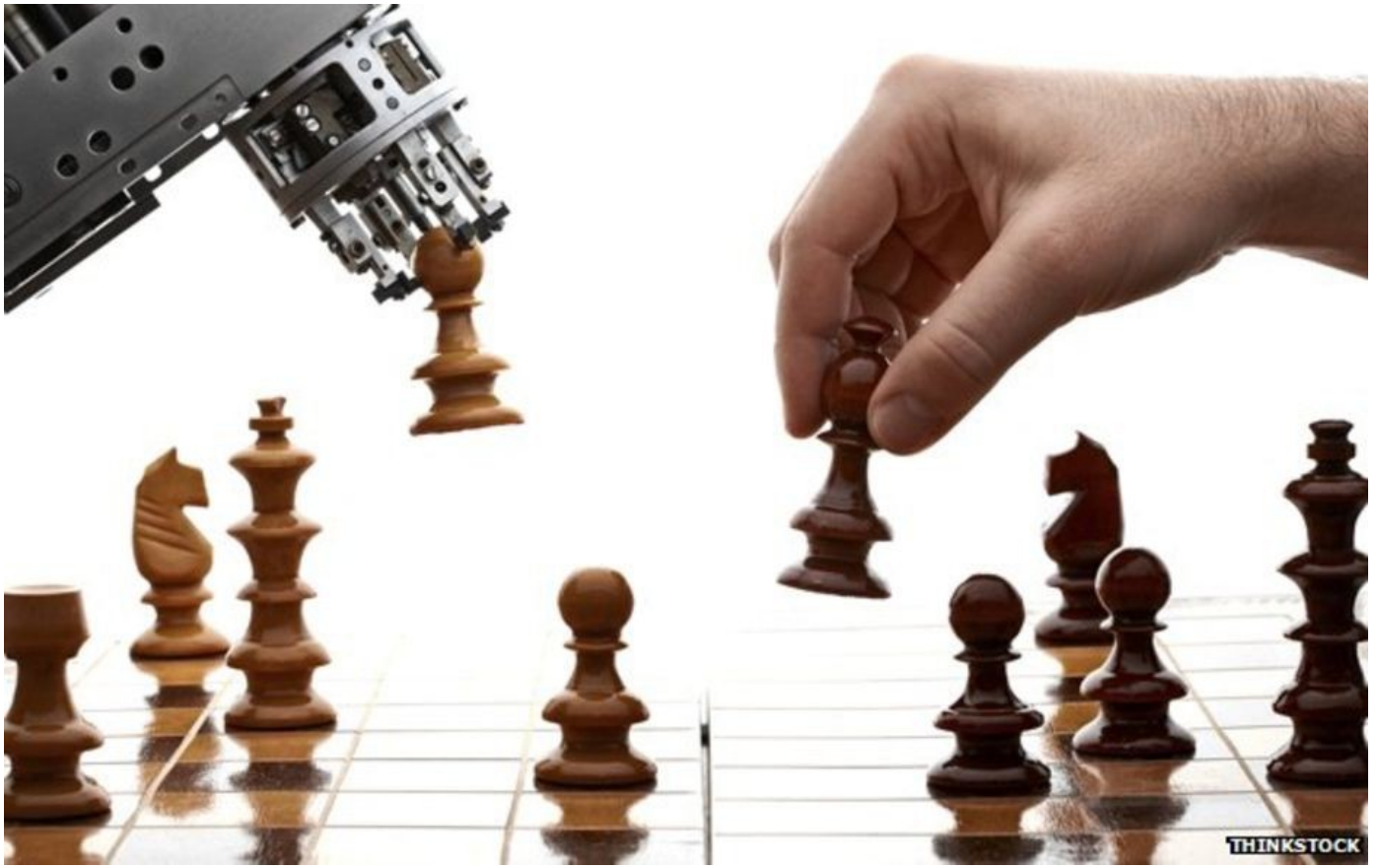


Introduction to Artificial Intelligence

COSC 4550 / COSC 5550

Professor Cheney

11/13/17

Stability Issues with Deep RL

Naive Q-learning oscillates or diverges with neural nets

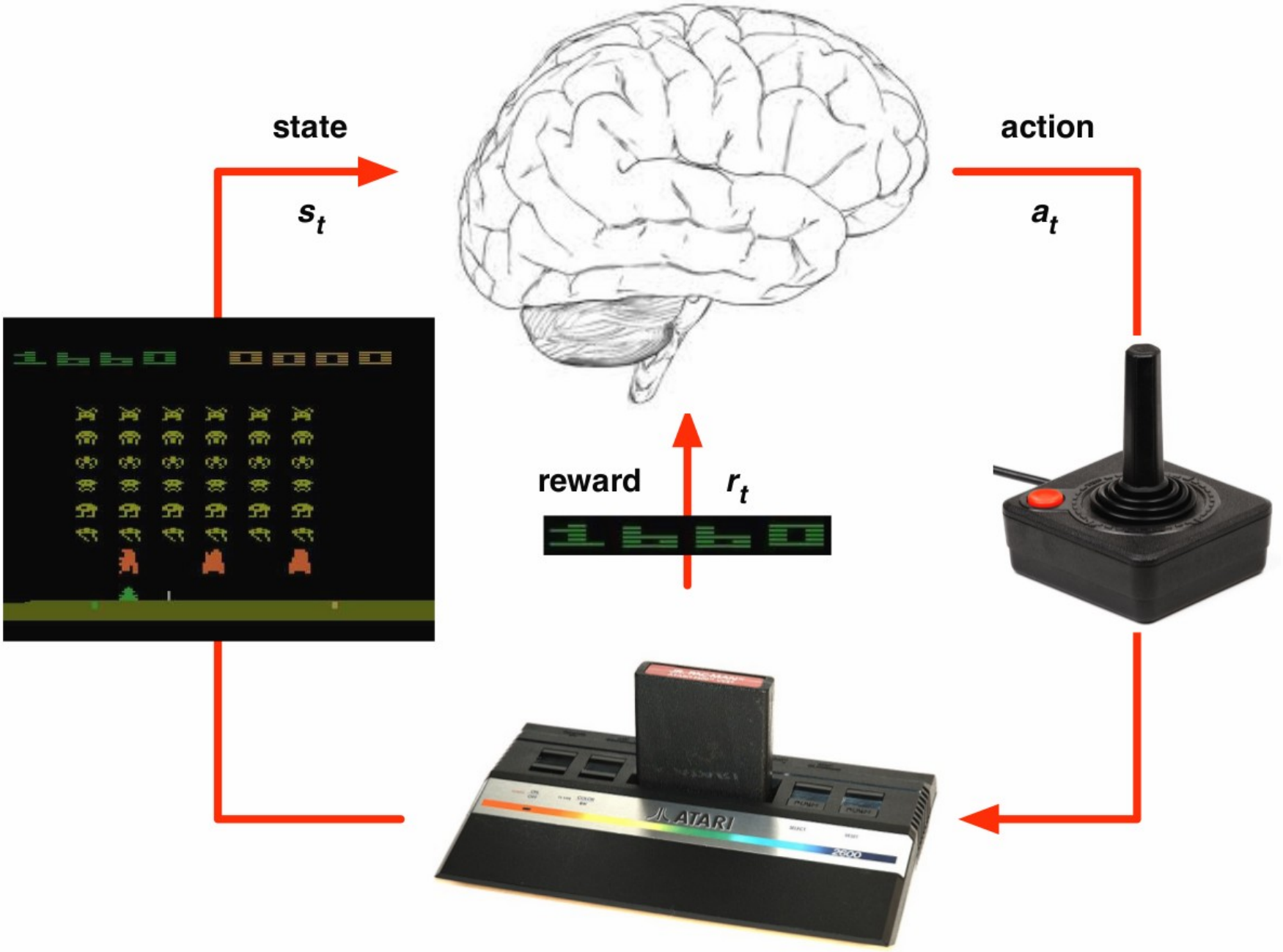
1. Data is sequential
   - Successive samples are correlated, non-iid
2. Policy changes rapidly with slight changes to Q-values
   - Policy may oscillate
   - Distribution of data can swing from one extreme to another
3. Scale of rewards and Q-values is unknown
   - Naive Q-learning gradients can be large
     unstable when backpropagated

Deep Q-Networks

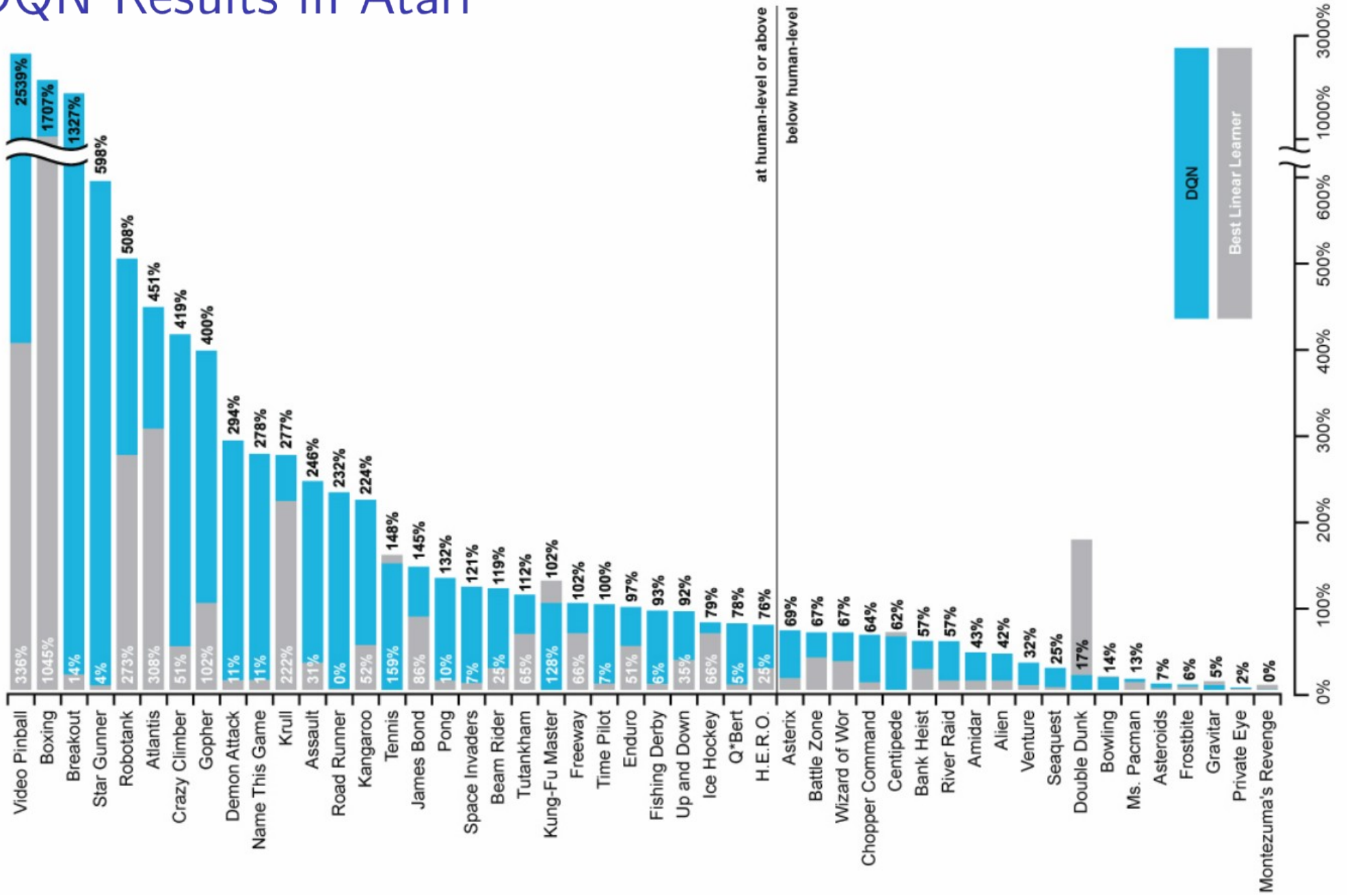
DQN provides a stable solution to deep value-based RL

1. Use experience replay
   - Break correlations in data, bring us back to iid setting
   - Learn from all past policies
2. Freeze target Q-network
   - Avoid oscillations
   - Break correlations between Q-network and target
3. Clip rewards or normalize network adaptively to sensible range
   - Robust gradients

Reinforcement Learning in Atari



DQN Results in Atari



How much does DQN help?

DQN

	Q-learning	Q-learning + Target Q	Q-learning + Replay	Q-learning + Replay + Target Q
Breakout	3	10	241	**317**
Enduro	29	142	831	**1006**
River Raid	1453	2868	4103	**7447**
Seaquest	276	1003	823	**2894**
Space Invaders	302	373	826	**1089**

learning policy networks

Deterministic Policy Gradient for Continuous Actions

- Represent deterministic policy by deep network $a = \pi(s, u)$ with weights u
- Define objective function as total discounted reward

$$J(u) = \mathbb{E}\left[r_1 + \gamma r_2 + \gamma^2 r_3 + ...\right]$$

- Optimise objective end-to-end by SGD

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

  - Update policy in the direction that most improves Q
  - i.e. Backpropagate critic through actor

Deterministic Actor-Critic

Use two networks: an actor and a critic

- Critic estimates value of current policy by Q-learning

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

- Actor updates policy in direction that improves Q

$$\frac{\partial J(u)}{\partial u} = \mathbb{E}_s\left[\frac{\partial Q(s, a, w)}{\partial a}\frac{\partial \pi(s, u)}{\partial u}\right]$$

Deterministic Deep Actor-Critic

- Naive actor-critic oscillates or diverges with neural nets
- DDAC provides a stable solution

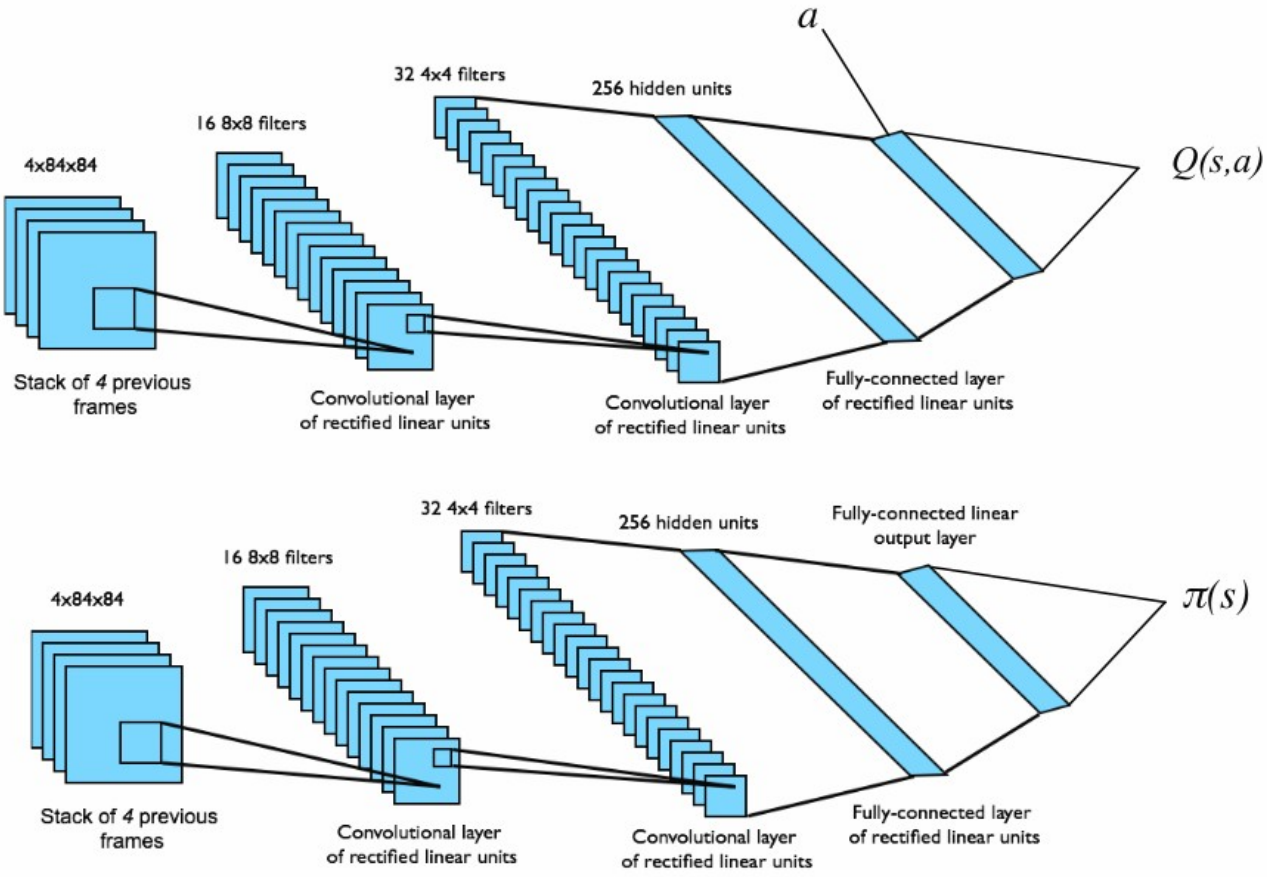
1. Use experience replay for both actor and critic
2. Use target Q-network to avoid oscillations

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

$$\frac{\partial J(u)}{\partial u} = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}} \left[\frac{\partial Q(s, a, w)}{\partial a} \frac{\partial \pi(s, u)}{\partial u} \right]$$

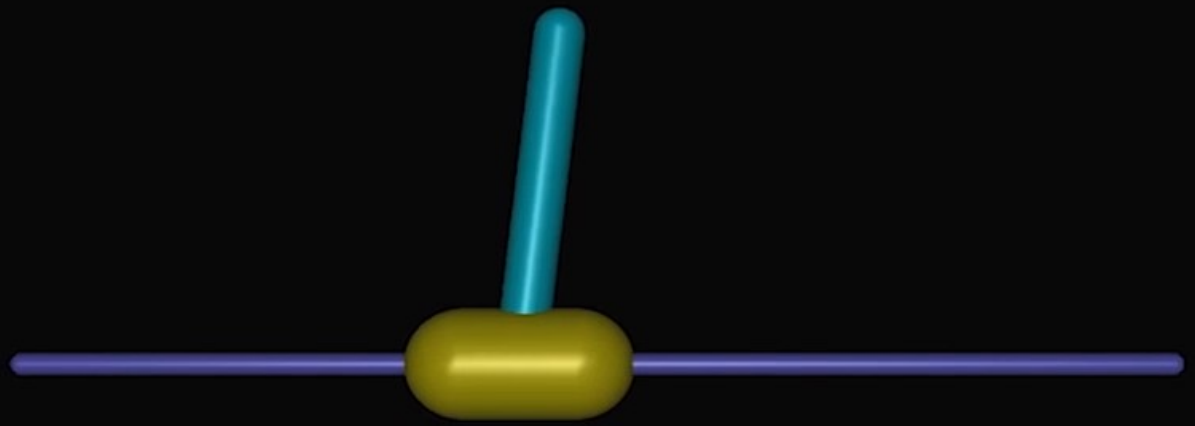
DDAC for Continuous Control

- End-to-end learning of control policy from raw pixels s
- Input state s is stack of raw pixels from last 4 frames
- Two separate convnets are used for Q and π
- Physics are simulated in MuJoCo



[Lillicrap et al.]


62fps
0
1
100
122



Emergence of Locomotion Behaviours in Rich Environments



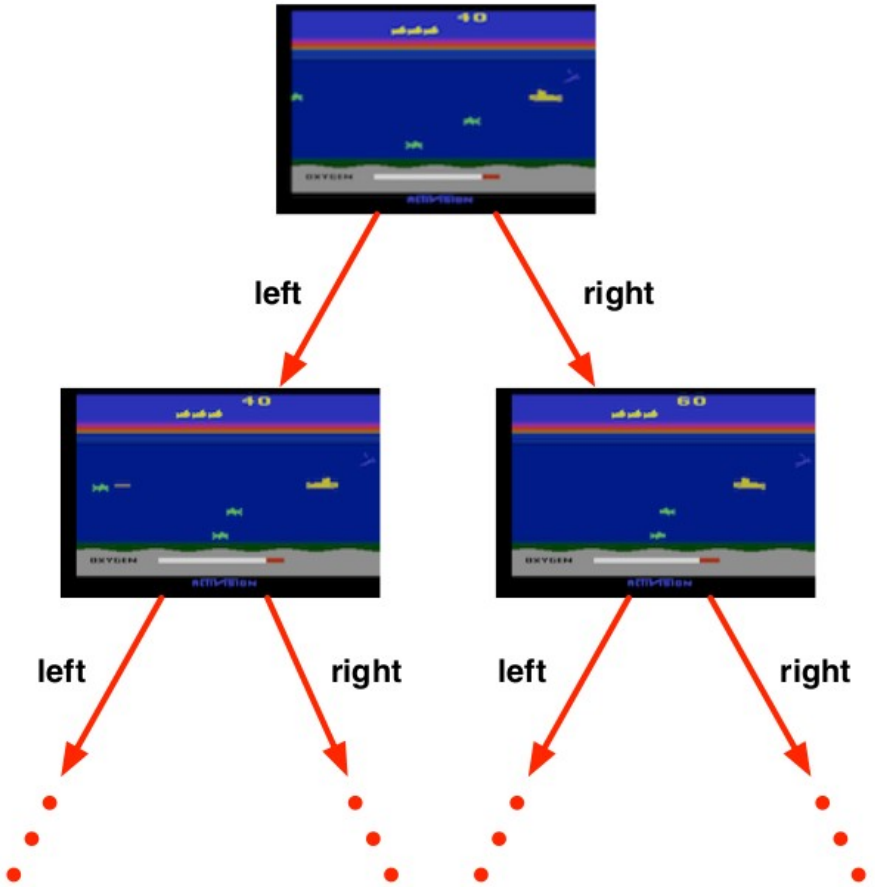
Model-Based RL

Learn a transition model of the environment

$$p(r, s' \mid s, a)$$

Plan using the transition model

- e.g. Lookahead using transition model to find optimal actions



Deep Models

- Represent transition model $p(r, s' \mid s, a)$ by deep network
- Define objective function measuring goodness of model
- e.g. number of bits to reconstruct next state (Gregor et al.)
- Optimise objective by SGD

Challenges of Model-Based RL

Compounding errors

- Errors in the transition model compound over the trajectory
- By the end of a long trajectory, rewards can be totally wrong
- Model-based RL has failed (so far) in Atari

Deep networks of value/policy can "plan" implicitly

- Each layer of network performs arbitrary computational step
- n-layer network can "lookahead" n steps
- Are transition models required at all?

Deep Learning in Go

Monte-Carlo search

- Monte-Carlo search (MCTS) simulates future trajectories
- Builds large lookahead search tree with millions of positions
- State-of-the-art 19×19 Go programs use MCTS
- e.g. First strong Go program *MoGo*

(Gelly et al.)

Convolutional Networks

- 12-layer convnet trained to predict expert moves
- Raw convnet (looking at 1 position, no search at all)
- Equals performance of MoGo with 10^5 position search tree

(Maddison et al.)

Program	Accuracy
Human 6-dan	$\sim$ 52%
12-Layer ConvNet	55%
8-Layer ConvNet*	44%
Prior state-of-the-art	31-39%

Program	Winning rate
GnuGo	97%
MoGo (100k)	46%
Pachi (10k)	47%
Pachi (100k)	11%

*Clarke & Storkey

ARTICLE

doi:10.1038/nature16961

Mastering the game of Go with deep neural networks and tree search

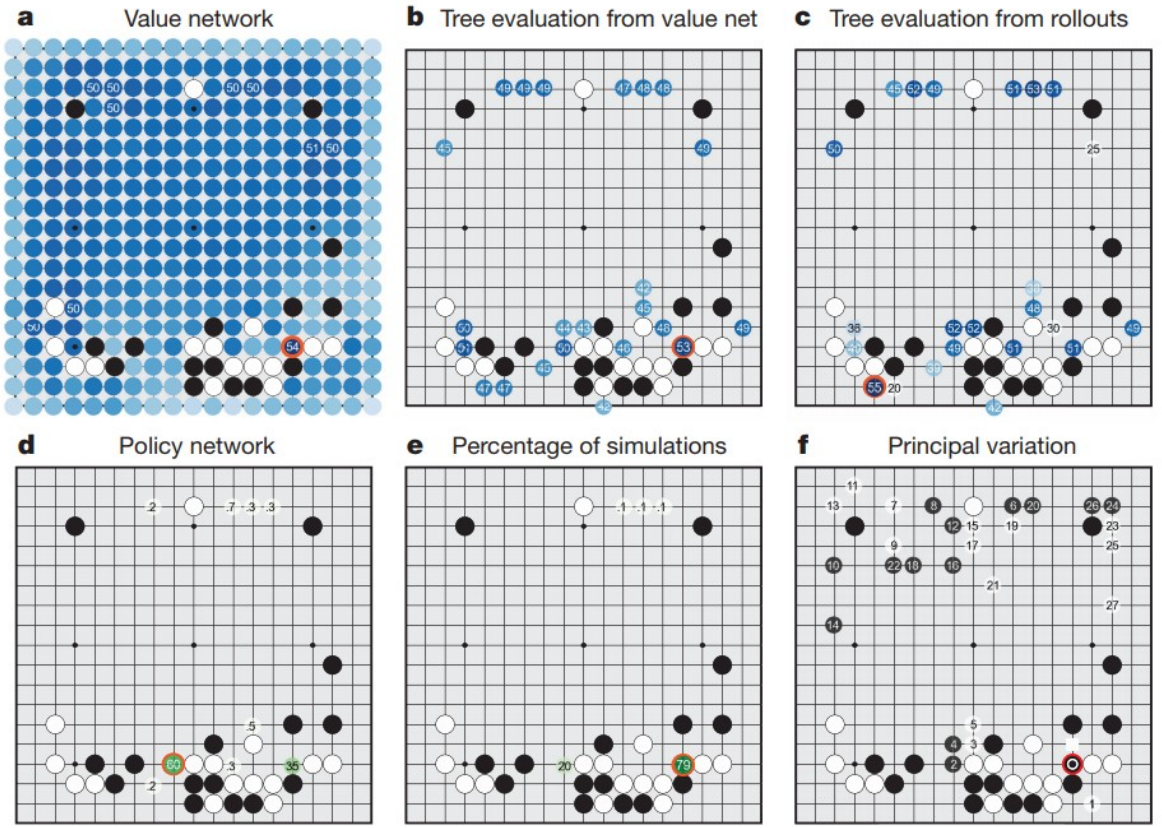


Figure 5 | How AlphaGo (black, to play) selected its move in an informal game against Fan Hui. For each of the following statistics, the location of the maximum value is indicated by an orange circle. **a**, Evaluation of all successors s' of the root position s, using the value network $v_\theta(s')$; estimated winning percentages are shown for the top evaluations. **b**, Action values $Q(s, a)$ for each edge (s, a) in the tree from root position s; averaged over value network evaluations only ($\lambda = 0$). **c**, Action values $Q(s, a)$, averaged over rollout evaluations only ($\lambda = 1$). **d**, Move probabilities directly from the SL policy network, $p_\sigma(a|s)$; reported as a percentage (if above 0.1%). **e**, Percentage frequency with which actions were selected from the root during simulations. **f**, The principal variation (path with maximum visit count) from AlphaGo's search tree. The moves are presented in a numbered sequence. AlphaGo selected the move indicated by the red circle; Fan Hui responded with the move indicated by the white square; in his post-game commentary he preferred the move (labelled 1) predicted by AlphaGo.

so you can learn a policy directly from pixels
instead of from state features or a game tree model

what's the big deal…?
my pacman simulator gives me those for free anyways

robotics!


fetch
robotics


r3d12
r3d24
r3d24
r3d32
r3d32


QUADCOPTER
NAVIGATION IN THE FOREST
TRAIL FOLLOWING UNDER THE TREE CANOPY

p.s. this is a great example/template
for you final project videos, they:

1) outline why their problem is important/hard

2) show how they collected data

3) say what model they trained

4) gave quantitative results

5) showed qualitative results (action shots!)