Recall off-policy learning involves two policies

- One policy π whose value function we are learning
    - the *target policy*
- Another policy μ that is used to select actions
    - the *behavior policy*

Off-policy is much harder with Function Approximation

- Even linear FA
- Even for prediction (two fixed policies π and μ)
- Even for Dynamic Programming
- The deadly triad: FA, TD, off-policy
    - Any two are OK, but not all three
    - With all three, we may get instability (elements of $\boldsymbol{\theta}$ may increase to $\pm\infty$)

The deadly triad

- The danger of divergence arises whenever we combine three things:

1. Function approximation
    - significantly generalizing from large numbers of examples
2. Bootstrapping
    - learning value estimates from other value estimates, as in dynamic programming and TD learning
3. Off-policy learning (Why is dynamic programming off-policy?)
    - learning about a policy from data not due to that policy, as in Q-learning, expected sarsa, tree backup

Any 2 Ok!

Final exam study guide

The final is comprehensive, covering chapters 3-10 and 12 of the text plus blind and heuristic search as covered in class and the readings. For review, see the practice questions for the final exam (and for the midterm), and all the written assignments. Also, be sure to do all the alpha-beta practice exercises.

The student who is knowledgeable of the following topics will do well on the exam:

the basic different kinds of search - what they are and what they are good for:
- breadth-first search
- depth-first search
- iterative-deepening search
- A* algorithm
- admissible heuristics
- minimax search
- alpha-beta pruning (do the exercises handed out, left-to-right then right-to-left)
- idea of Monte Carlo tree search

Markov decision processes
- optimal policies
- returns, discounting
- value functions (four of them - definition, uniqueness)
- Bellman equations as systems of linear equations

Dynamic programming
- value iteration
- policy iteration
- Generalized policy iteration

TD learning algorithms
- TD(0)
- Sarsa
- Expected Sarsa
- Q-learning

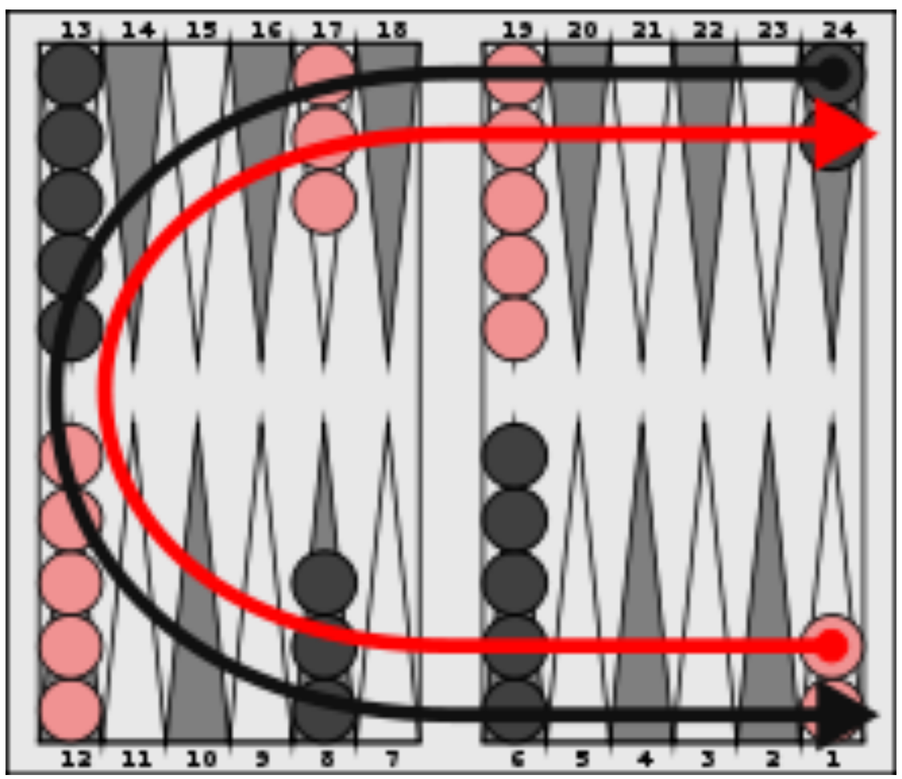
Eligibility traces
Linear function approximation with eligibility traces
n-step methods, the tree-backup algorithm
Tile coding
Monte Carlo learning
backup diagrams
- for TD, DP, MC algorithms, multi-step backups, and for each of the 4 value functions
- how to draw the diagram for each kind of method
- how to write the backup equation for each diagram

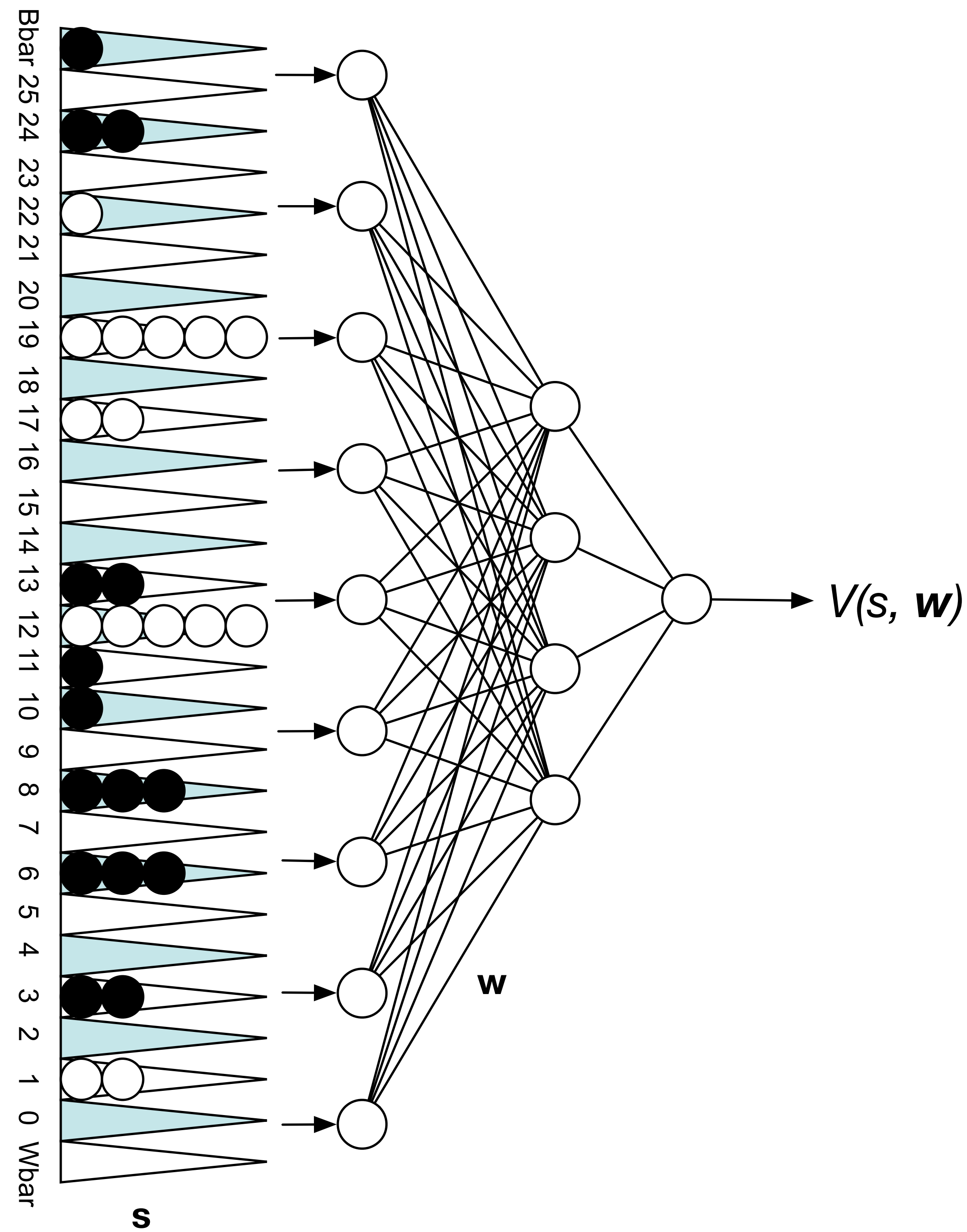
TD vs MC, batch updating, and the MSE
The role of lambda in TD(lambda)
incremental computation of averages
Dyna
The difference between planning and learning

Deep Learning
= Neural Networks

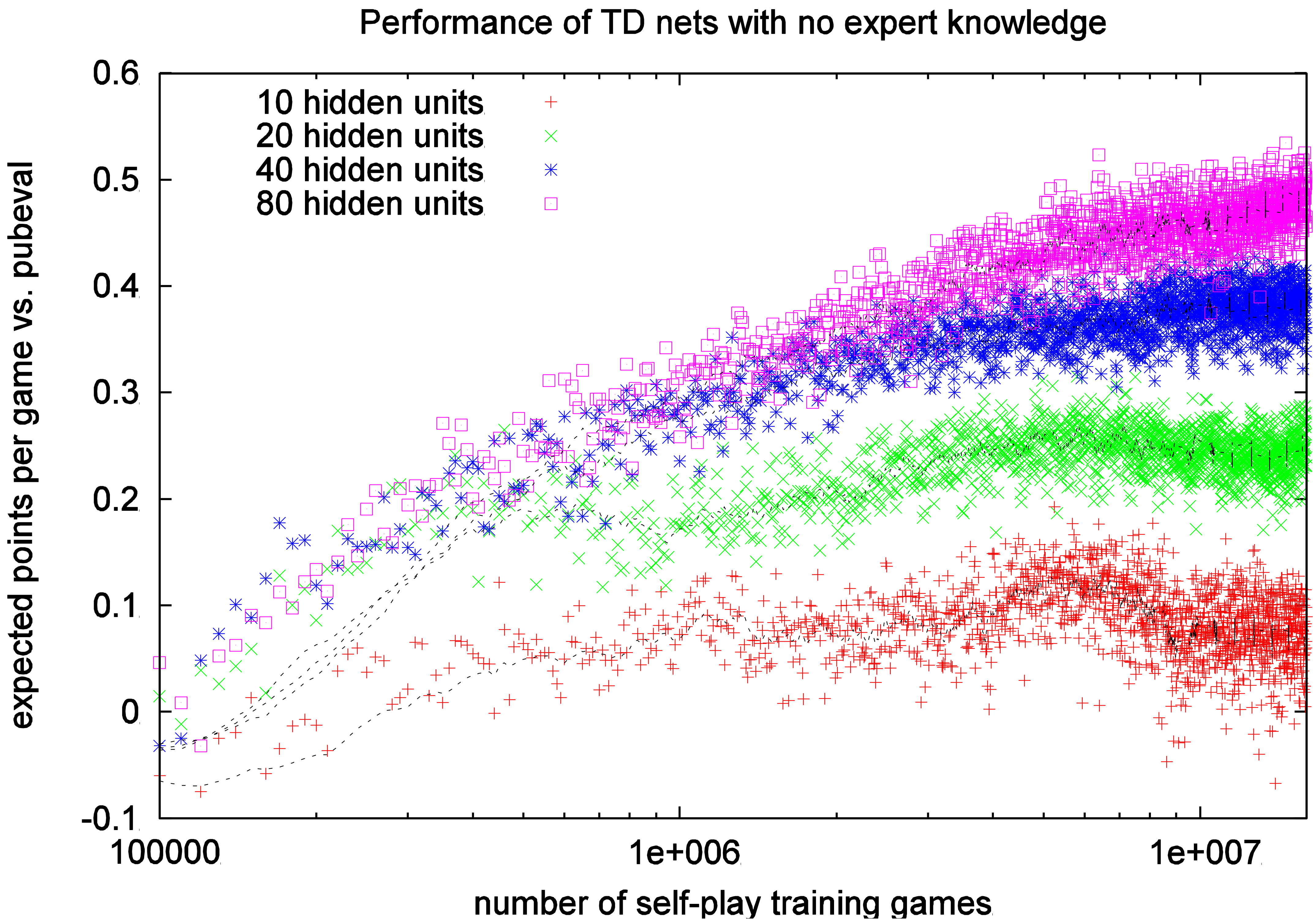
Deep RL = RL + NN

Example: TD Gammon



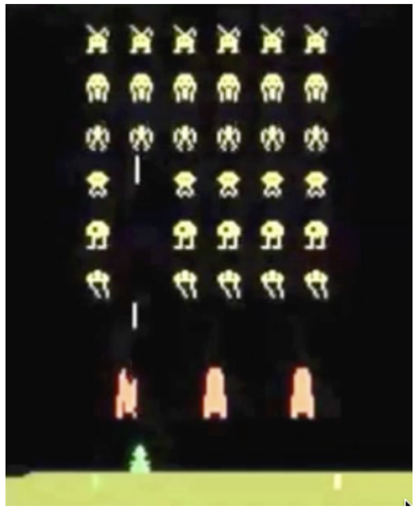


New TD-Gammon Results



RL + DL applied to Classic Atari Games

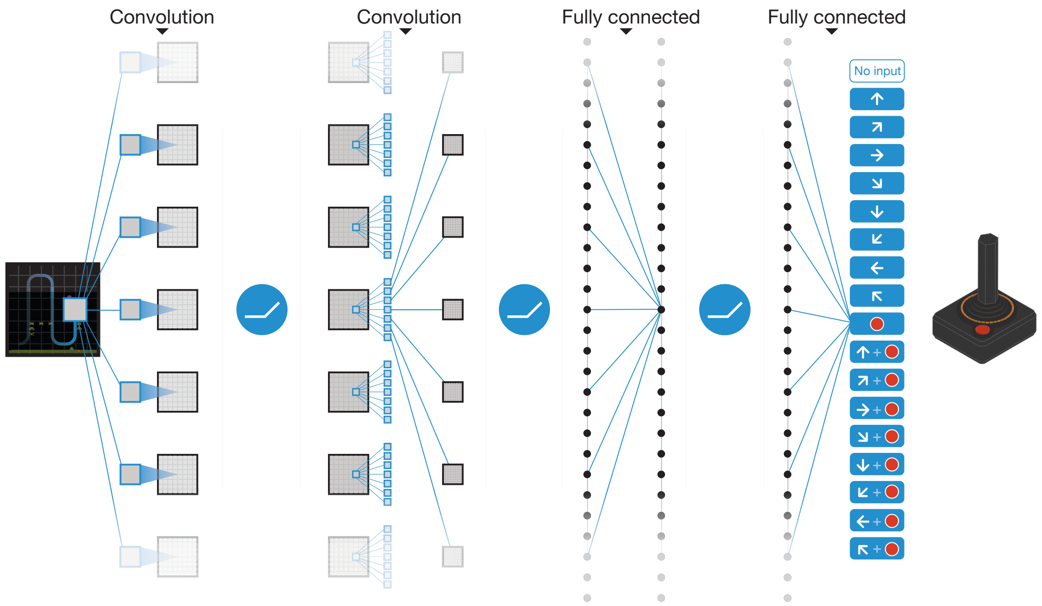
Google Deepmind 2015, Bowling et al. 2012





- Learned to play 49 games for the Atari 2600 game console, without labels or human input, from self-play and the score alone

mapping raw screen pixels



to predictions of final score for each of 18 joystick actions

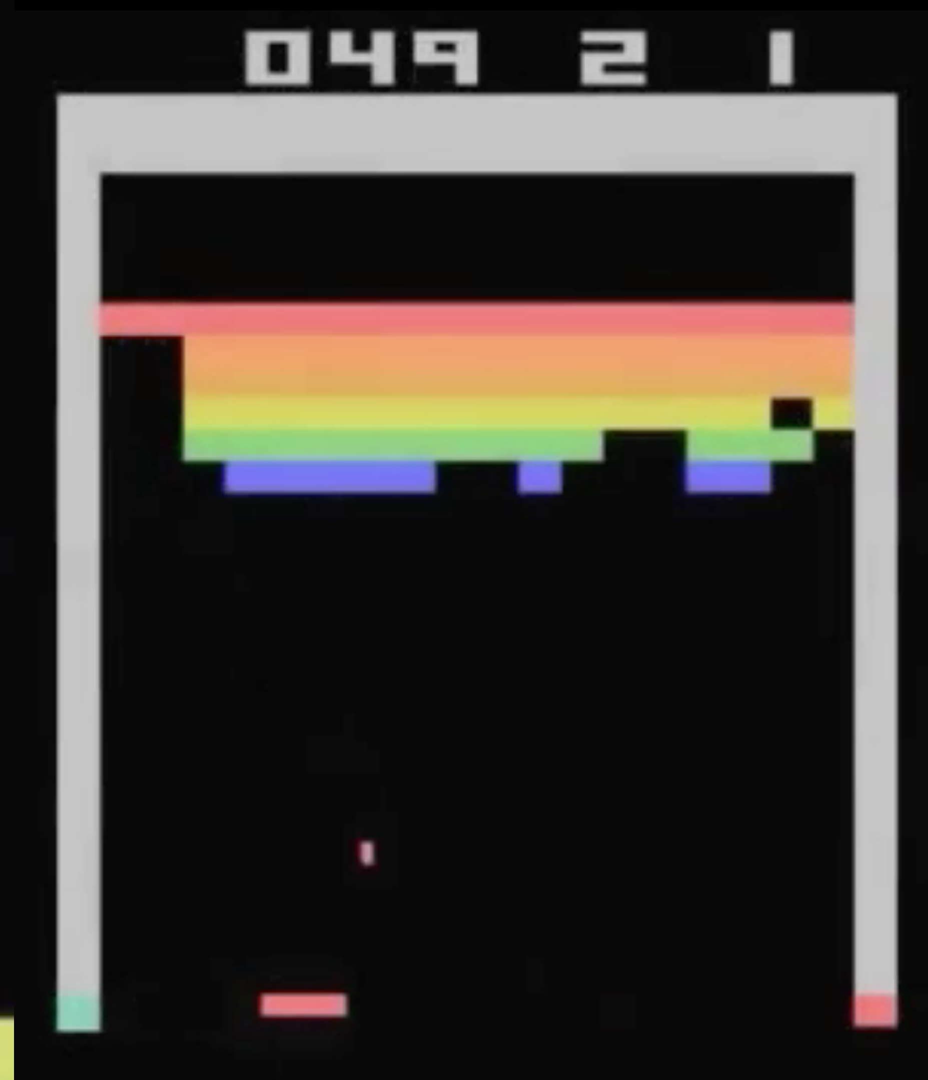
- Learned to play better than all previous algorithms and at human level for more than half the games

Same learning algorithm applied to all 49 games! No human tuning

RL + DL Performance on Atari Games

(Deep Learning)

Space Invaders



Breakout

Enduro

The University of Alberta connection

the lead researchers were UofA alumni

doi:10.1038/nature14236

Human-level control through deep reinforcement learning

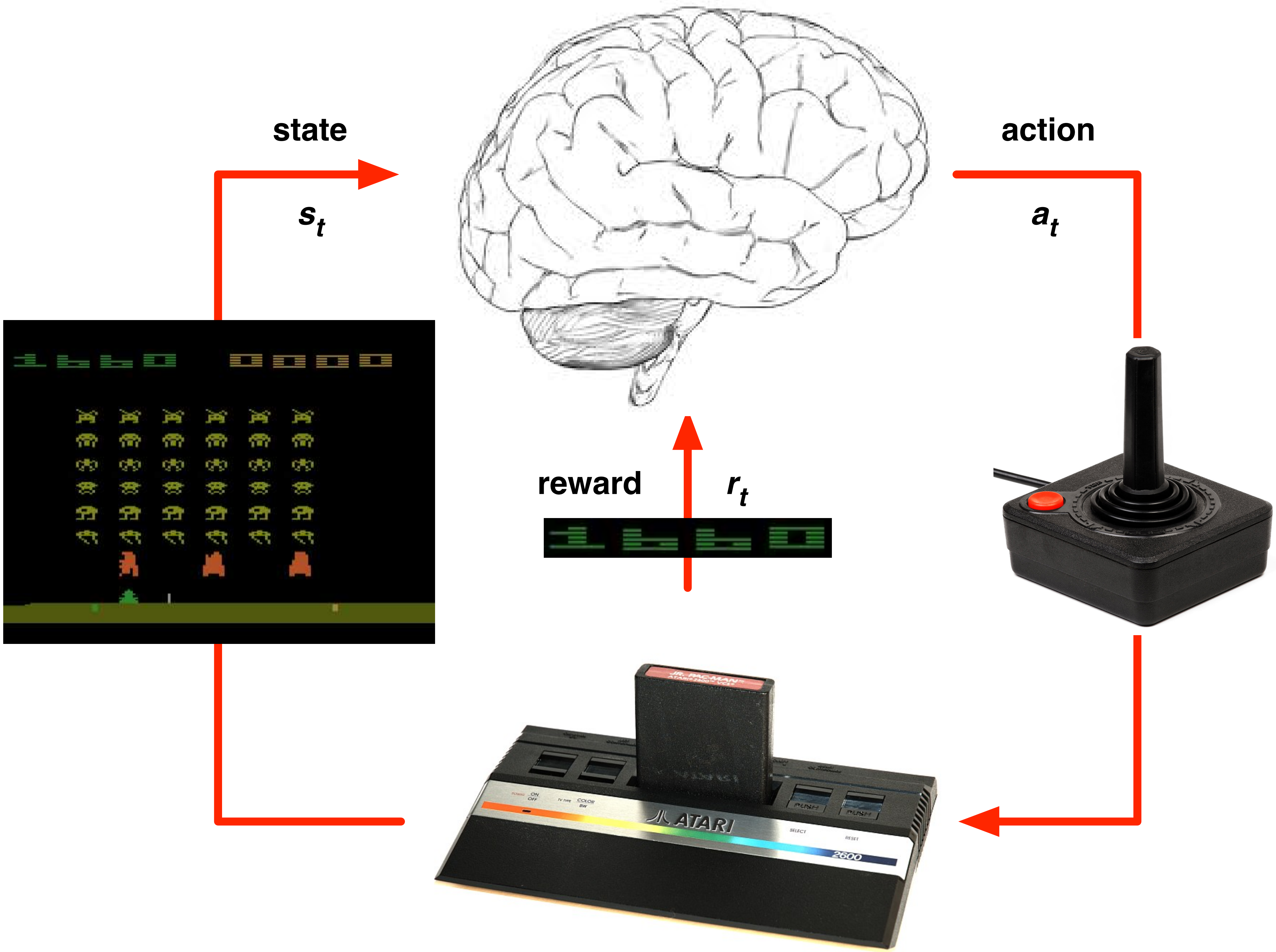
UofA CS MSc, supervised by Csaba Szepesvari

UofA CS PhD, supervised by Martin Mueller and me

Volodymyr Mnih[1]*, Koray Kavukcuoglu[1]*, David Silver[1]*, Andrei A. Rusu[1], Joel Veness[1], Marc G. Bellemare[1], Alex Graves[1], Martin Riedmiller[1], Andreas K. Fidjeland[1], Georg Ostrovski[1], Stig Petersen[1], Charles Beattie[1], Amir Sadik[1], Ioannis Antonoglou[1], Helen King[1], Dharshan Kumaran[1], Daan Wierstra[1], Shane Legg[1] & Demis Hassabis[1]

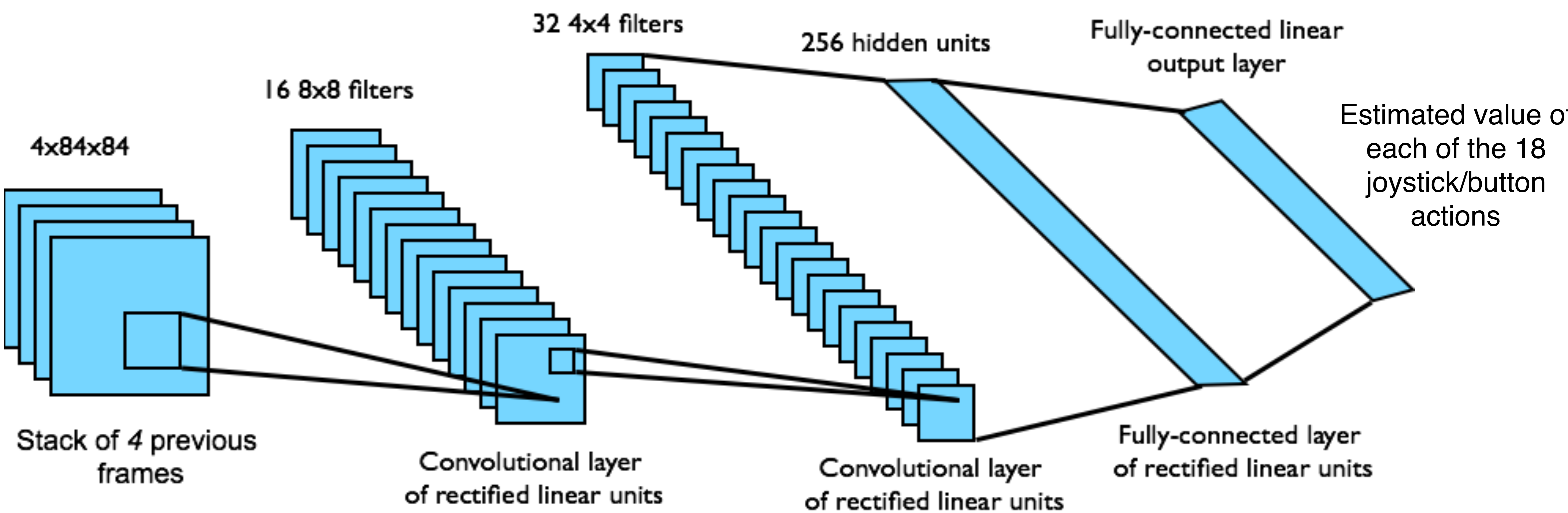
UofA CS PDF, PhD, supervised by Mike Bowling

Reinforcement Learning in Atari



DQN in Atari

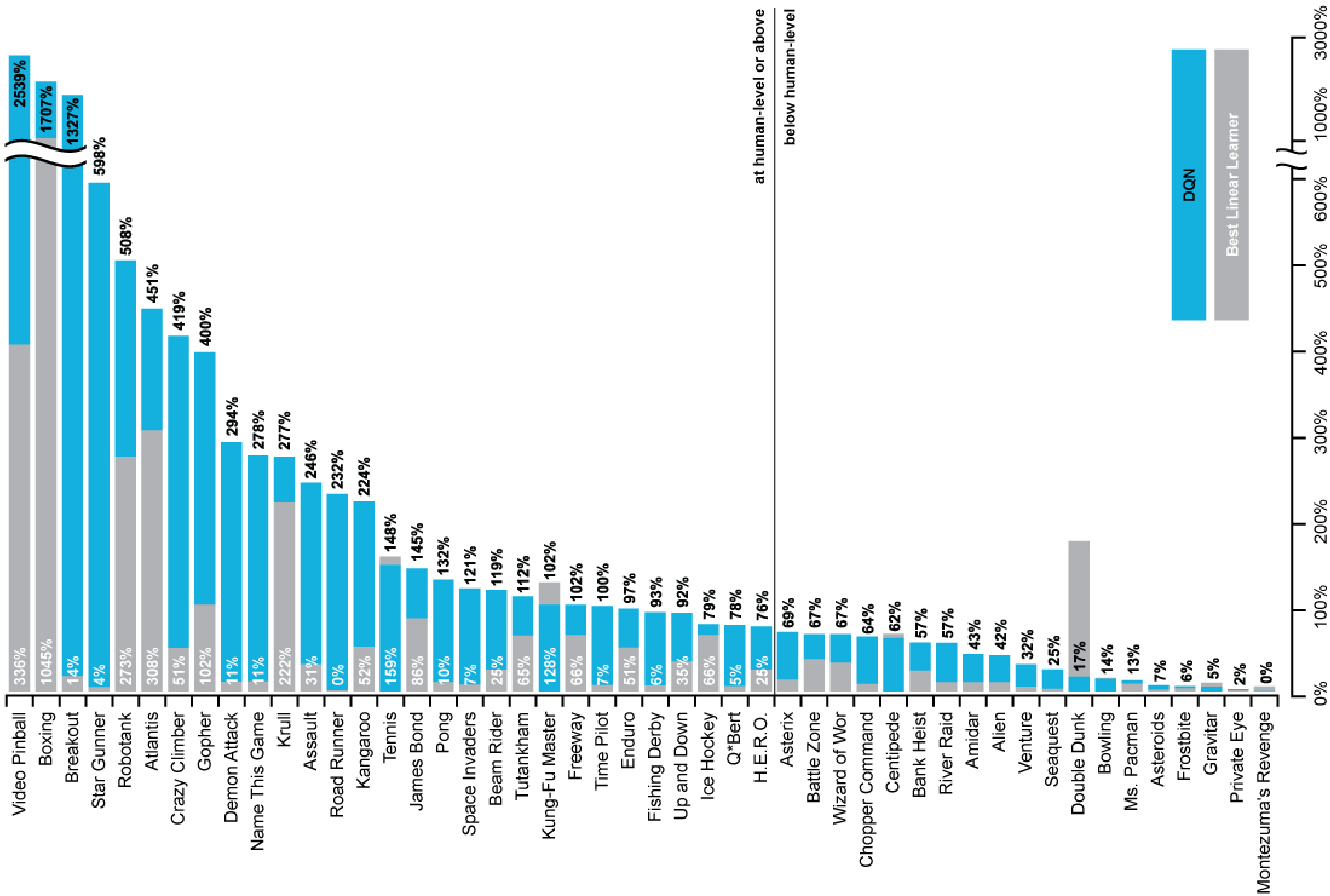
- End-to-end learning of values $Q(s, a)$ from pixels s
- Input state s is stack of raw pixels from last 4 frames
- Output is $Q(s, a)$ for 18 joystick/button positions
- Reward is change in score for that step



Network architecture and hyperparameters fixed across all games

[Mnih et al.]

DQN Results in Atari




AlphaGo