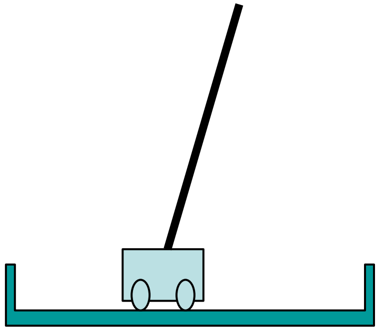
CS 486/686: Introduction to Artificial Intelligence

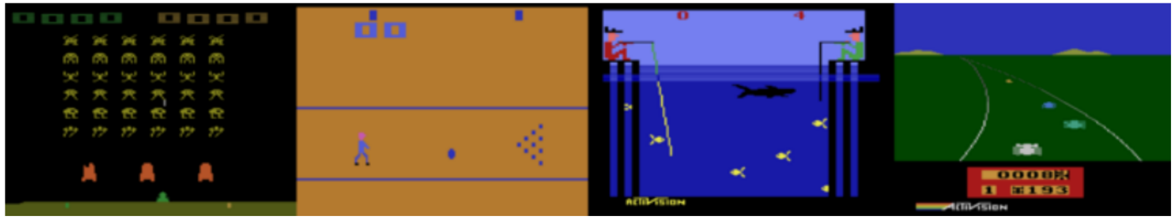
Reinforcement Learning

Large State Spaces

- Computer Go: 3^{361} states
- Inverted pendulum:
  - 4 dimensional, continuous state space



- Atari: 210 x 160 x 3 dimensions (pixel values)



Value-Function Approximation

- So far we have represented value functions by a look-up table (tabular RL)
  - Every state s has an entry V(S)
  - Every state-action pair s,a has an entry Q(s,a)

- Issue
  - There are too many states or actions to store in memory
  - It is too slow to learn the value of each state individually

Value-Function Approximation

- Estimate value functions with function approximation

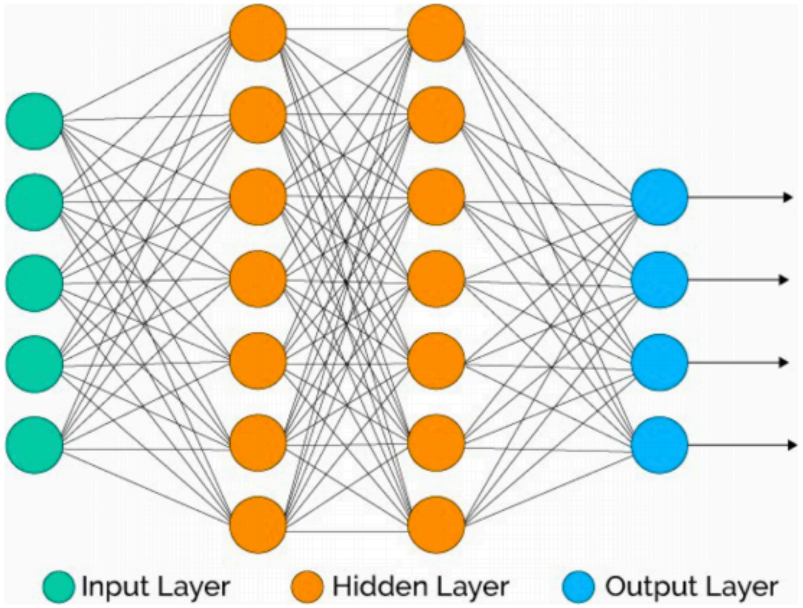
$$V^\pi(s) \sim \hat{V}(s, \mathbf{w})$$

$$Q^\pi(s, a) \sim \hat{Q}(s, a, \mathbf{w})$$

- Let $s=(x_1, x_2, \ldots, x_n)^T$ or $(s,a)=(x_1(s,a), \ldots, x_n(s,a))^T$
  - Linear: $V(s, \mathbf{w}) = \sum_i w_i \, x_i(s)$, $Q(s,a,\mathbf{w}) = \sum_i w_i \, x_i(s, a)$
  - Non-linear (e.g. neural networks): $V(s,\mathbf{w})$ $(Q(s,a,\mathbf{w}))= g(x;\mathbf{w})$

Recall: Neural Networks

- Network of units linked by weighted edges



- Each unit computes: $z=h(w^Tx+b)$
  - Inputs: x
  - Outputs: z
  - Weights: w
  - Bias: b
  - Activation function: h

- Neural networks with at least one "large enough" hidden layer consisting of sigmoid/tanh/Gaussian units can approximate any function arbitrarily closely

Gradient Q-Learning

- Minimize the error between Q-value estimate and a target
  - Estimate: Q(s,a,**w**)
  - Target: r+γmax$_{a'}$ Q(s',a',**w'**)

- Squared Error:

$$\mathrm{err}(\mathbf{w}) = \frac{1}{2}[Q(s,a,\mathbf{w}) - r - \gamma \max_{a'} Q(s',a',\mathbf{w}')]^2$$

- Gradient:

$$\frac{\partial \mathrm{Err}}{\partial \mathbf{w}} = [Q(s,a,\mathbf{w}) - r - \gamma \max_{a'} Q(s',a',\mathbf{w}')] \frac{\partial Q(s,a,\mathbf{w})}{\partial \mathbf{w}}$$

Gradient Q-Learning

Initialize weights $\boldsymbol{w}$ at random in $[-1,1]$

Observe current state s

Loop

Select action a and execute it

Receive immediate reward r

Observe new state s'

Gradient: $\frac{\partial Err}{\partial \boldsymbol{w}} = [Q_{\boldsymbol{w}}(s,a) - r - \gamma \max_{a'} Q_{\boldsymbol{w}}(s',a')] \frac{\partial Q_{\boldsymbol{w}}(s,a)}{\partial \boldsymbol{w}}$

Update weights: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \frac{\partial Err}{\partial \boldsymbol{w}}$

Update state: $s \leftarrow s'$

Convergence?

- Tabular Q-learning converges when

$$\sum_{t=0}^{\infty} \alpha_t = \infty \text{ and } \sum_{t=0}^{\infty} \alpha_t^2 < \infty$$

- We typically set α_t(s,a)=1/n(s,a) where n(s,a) is the number of times (s,a) is visited or 1/t.

- Linear function approximation
  - Same convergence guarantees
- Non-linear function approximation
  - No convergence guarantee

Non-linear function approximation

Handling divergence:

- Experience replay
- Use two networks:
  - Q-network
  - Target network

Experience Replay

- Store previous experiences (s,a,,s’,r) into a buffer and sample a mini-batch of previous experiences at each step to learn by Q-learning
  - Break correlations between successive updates (more stable learning)
  - Less interactions with environment needed (better data efficiency)

Target Network

- Use a target network that is updated only occasionally

- Repeat for each (s,a,s',r) in a mini-batch:

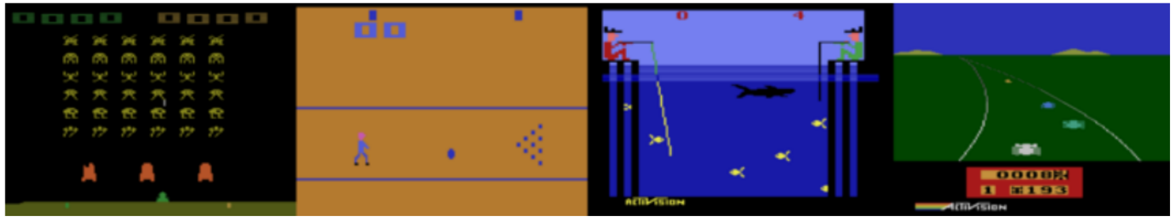
$$\mathbf{w} \leftarrow \mathbf{w} - \alpha_t[Q(s,a,\mathbf{w}) - r - \gamma \max_{a'} Q(s',a',\mathbf{w}')] \frac{\partial Q(s,a,\mathbf{w})}{\partial \mathbf{r}}$$

$$\mathbf{w}' \leftarrow \mathbf{w}$$

- Observe: similar in spirit to value iteration

Deep Q-network (DQN)

- Gradient Q-learning with
  - Deep neural networks
  - Experience replay
  - Target network



Playing Atari with Deep Reinforcement Learning

Volodymyr Mnih Koray Kavukcuoglu David Silver Alex Graves Ioannis Antonoglou

Daan Wierstra Martin Riedmiller

DeepMind Technologies

{vlad,koray,david,alex.graves,ioannis,daan,martin.riedmiller} @ deepmind.com

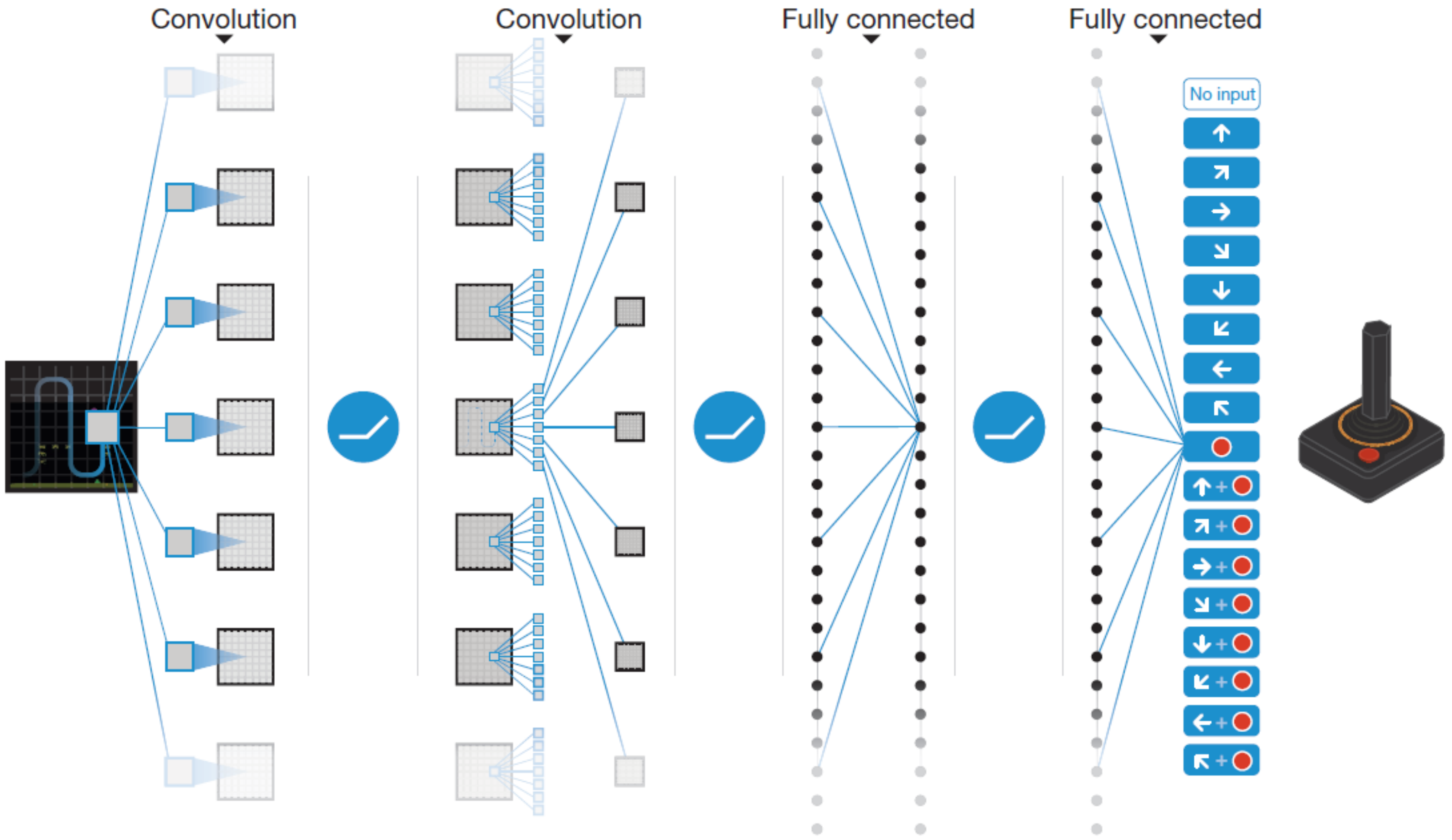
Abstract

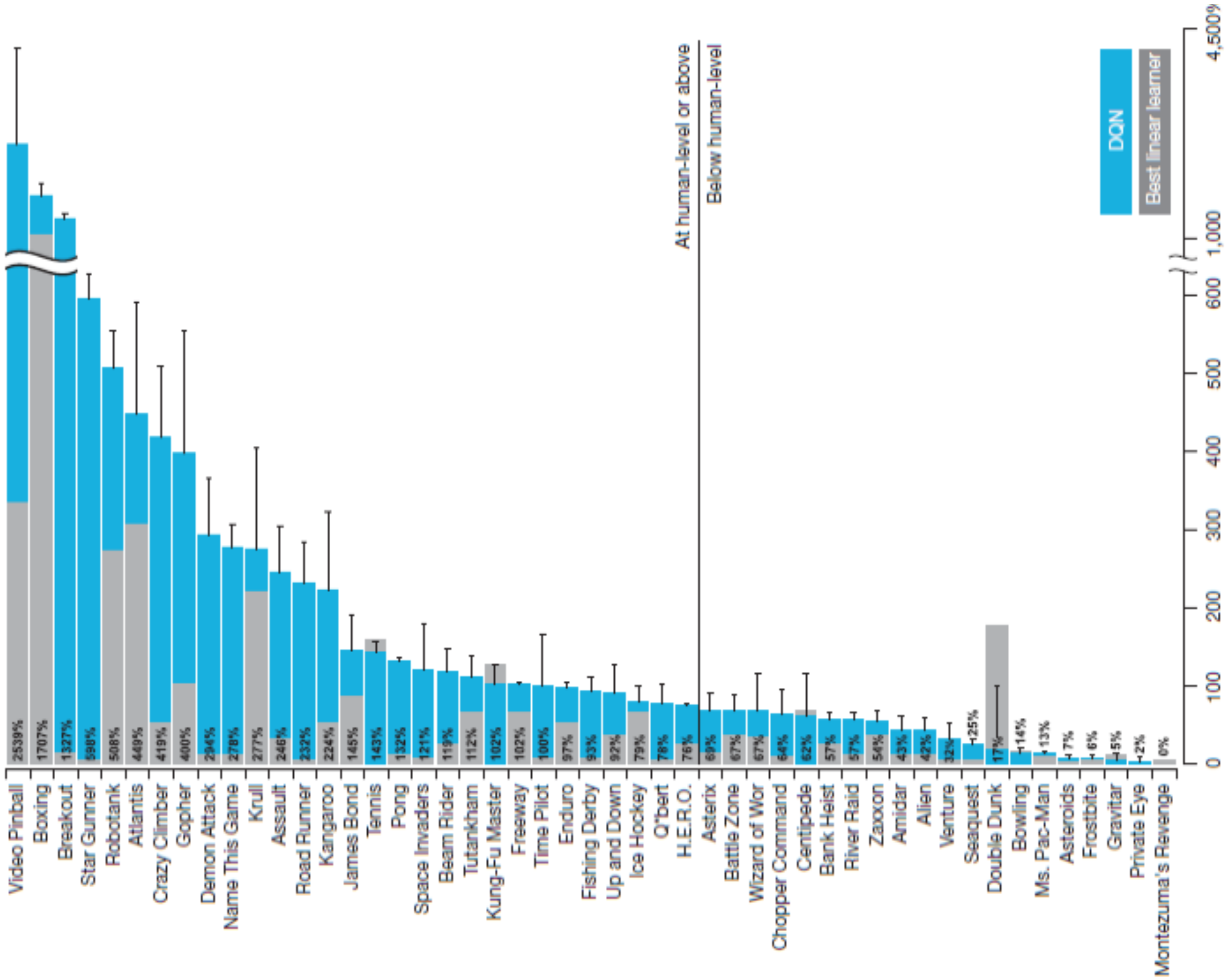
We present the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning. The model is a convolutional neural network, trained with a variant of Q-learning, whose input is raw pixels and whose output is a value function estimating future rewards. We apply our method to seven Atari 2600 games from the Arcade Learning Environment, with no adjustment of the architecture or learning algorithm. We find that it outperforms all previous approaches on six of the games and surpasses a human expert on three of them.

DQN

Initialize weights $\boldsymbol{w}$ and $\overline{\boldsymbol{w}}$ at random in $[-1,1]$
Observe current state s
Loop
    Select action a and execute it
    Receive immediate reward r
    Observe new state s'
    Add $\langle s, a, s', r \rangle$ to experience buffer
    Sample mini-batch of experiences from buffer
    For each experience $\langle \hat{s}, \hat{a}, \hat{s}', \hat{r} \rangle$ in mini-batch
        Gradient: $\frac{\partial Err}{\partial \boldsymbol{w}} = [Q_{\boldsymbol{w}}(\hat{s}, \hat{a}) - \hat{r} - \gamma \max_{\hat{a}'} Q_{\overline{\boldsymbol{w}}}(\hat{s}', \hat{a}')] \frac{\partial Q_{\boldsymbol{w}}(\hat{s}, \hat{a})}{\partial \boldsymbol{w}}$
        Update weights: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \frac{\partial Err}{\partial \boldsymbol{w}}$
    Update state: $s \leftarrow s'$
    Every c steps, update target: $\overline{\boldsymbol{w}} \leftarrow \boldsymbol{w}$

DQN for Atari




DQN
Best linear learner
At human-level or above
Below human-level
0
100
200
300
400
500
600
1,000
4,500%
Video Pinball 2539%
Boxing 1707%
Breakout 1327%
Star Gunner 598%
Robotank 508%
Atlantis 449%
Crazy Climber 419%
Gopher 400%
Demon Attack 294%
Name This Game 278%
Krull 277%
Assault 246%
Road Runner 232%
Kangaroo 224%
James Bond 145%
Tennis 143%
Pong 132%
Space Invaders 121%
Beam Rider 119%
Tutankham 112%
Kung-Fu Master 102%
Freeway 102%
Time Pilot 100%
Enduro 97%
Fishing Derby 93%
Up and Down 92%
Ice Hockey 79%
Q*bert 78%
H.E.R.O. 76%
Asterix 69%
Battle Zone 67%
Wizard of Wor 67%
Chopper Command 64%
Centipede 62%
Bank Heist 57%
River Raid 57%
Zaxxon 54%
Amidar 43%
Alien 42%
Venture 32%
Seaquest 25%
Double Dunk 17%
Bowling 14%
Ms. Pac-Man 13%
Asteroids 7%
Frostbite 6%
Gravitar 5%
Private Eye 2%
Montezuma's Revenge 0%