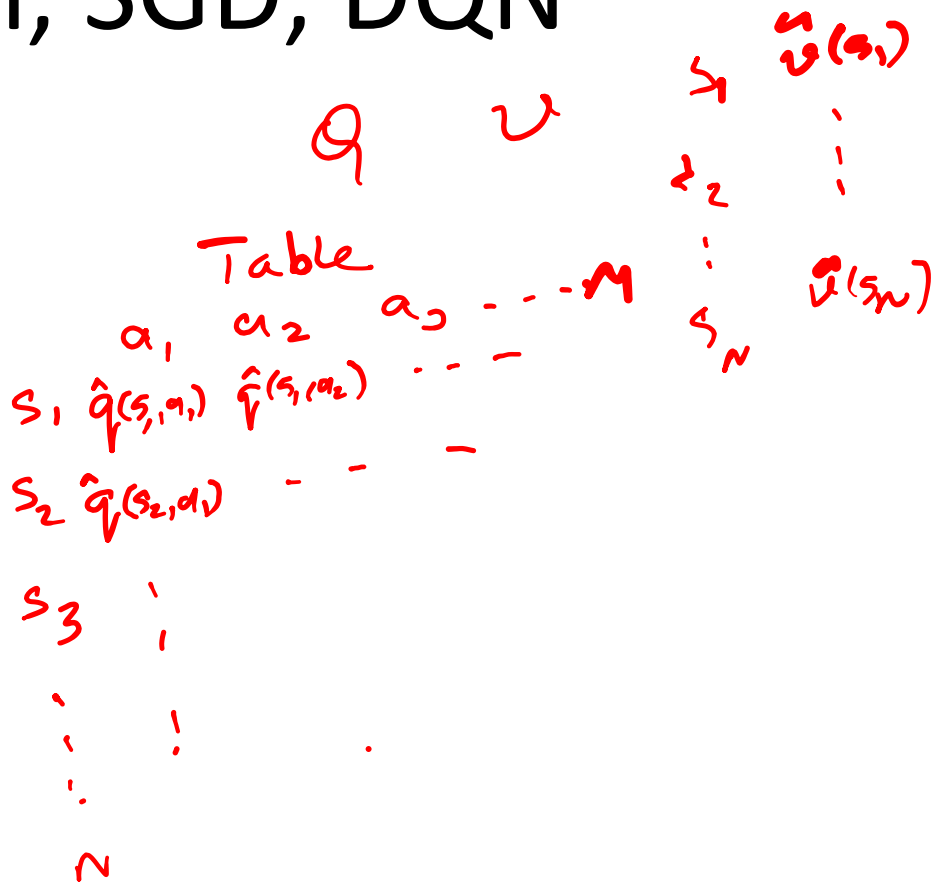
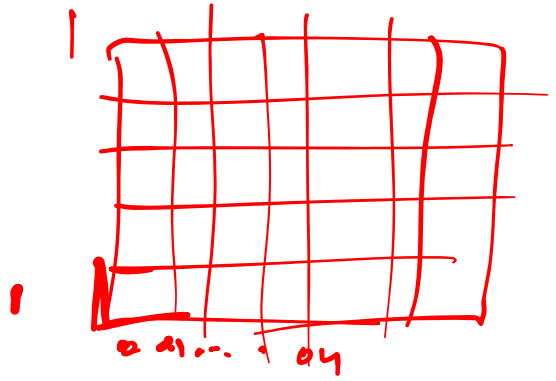
Function Approximation, SGD, DQN

B. Ravindran



Need for Function Approximation

- Issues with large state/action spaces:
  - tabular approaches not memory efficient
  - data sparsity
  - continuous state/action spaces
  - generalization
- Use a parameterized representation
  - Value Functions
  - Policies
  - Models $p(s', r \mid s, a)$



linear regression.

$f(x) = w_1 x_1 + w_2 x_2 + \cdots + w_p x_p$.

$f(x; w)$

$f(x; \theta)$

Value Function Approximation

- Least squares $Q(s_t, a_t) = f(s_t, a_t; w_t)$

$$w_{t+1} = w_t - \frac{1}{2}\alpha \nabla_{w_t} [q_*(s_t, a_t) - Q(s_t, a_t)]^2$$

- But we don't know the target!
- Use the TD *target.*

$$w_{t+1} = w_t - \frac{1}{2}\alpha \nabla_{w_t} \left[r_{t+1} + \gamma \max_a Q(s_t, a) - Q(s_t, a_t) \right]^2$$

Semi Gradient Methods

- While computing the gradient of the TD error in Q-learning, we typically ignore the gradient w.r.t the TD target. Hence, it is a **Semi Gradient** method i.e we are computing an approximation of the true gradient.

- Eg: **Linear Functions:**

$$Q(s_t, a_t) = \phi^T(s_t, a_t) \times w_t$$

$$\delta_t = r_{t+1} + \gamma \max_a Q(s_t, a) - Q(s_t, a_t) \quad \text{TD Error}$$

$$\nabla_{w_t} \left[r_{t+1} + \gamma \max_a Q(s_t, a) - Q(s_t, a_t) \right]^2 = -2\delta_t \phi(s_t, a_t)$$

$$w_{t+1} = w_t + \alpha\, \delta_t \phi(s_t, a_t)$$

Stochastic Gradient Descent

- In **Stochastic Gradient Descent**, the true gradient of the loss function is approximated by the gradient at a single example.
- In practice, we usually perform **Mini Batch Gradient Descent**, where we compute the gradient using a mini batch of examples. This allows for more efficient computation and smoother convergence.
    - Still SGD

Tile and Coarse coding

Can exploit the possibility that q values of near-by states wouldn't change a lot.
Assume the task is to learn a policy on a big gridworld.

Tabular Lookup!

Navie method:- State aggregation

Divide the grid into smaller 10x10 grid.
Abrupt change in Q-value of states that lie on the boundary.

Coarse coding:-

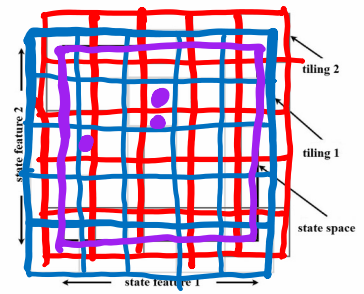
Avoids abrupt changes in the value of the state. Smoothens the qvalues during transition from one cluster to another.
Issue:-

No uniformity in the number of 'ON' bits used to represent a state.

Tile coding:-

Form of coarse coding but systematic.
Number of 'ON' bits == Number of tiles us

Randomly shift the tiling.



Non-Linear Function Approximator

1.Linear function approximators are ~~very~~ somewhat restrictive. Can only model linear functions. Basis expansion does help to generate non-linear functions in the original input space.

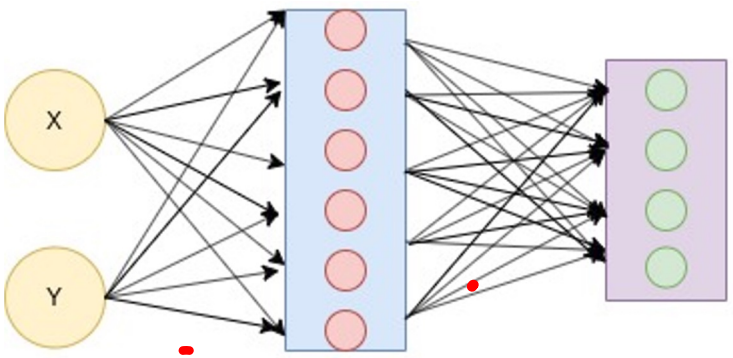
2.Non-linear approximators can model complex functions and are very powerful.

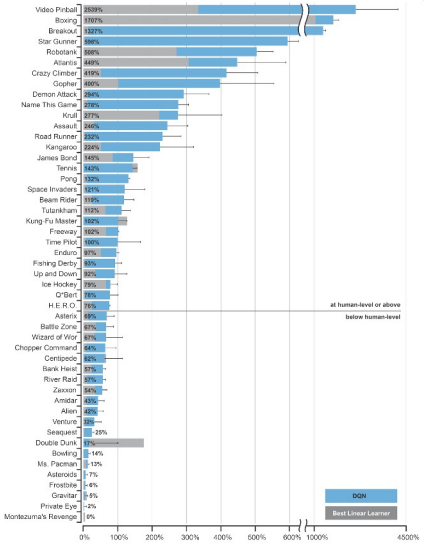
3.The features are learnt on the fly and are not hardcoded as is the case with tile and sparse coding.

4.Can generalize to unseen states.

Disadvantage:-

Requires a lot of data and compute.





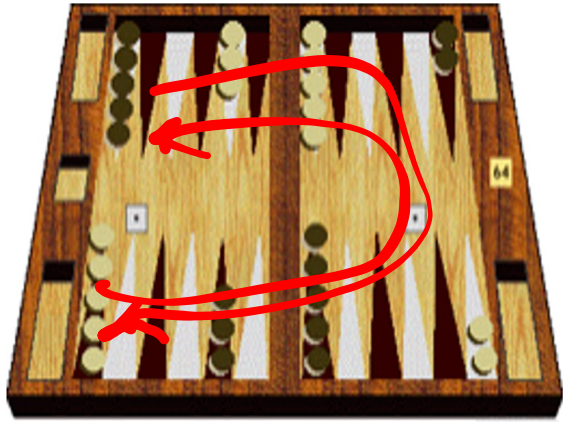
Human Level Backgammon player

TD-Gammon (Tesauro 92, 94, 95)

Beat the best human player in 1995

Learnt completely by *self play*

New moves not recorded by humans in centuries of play



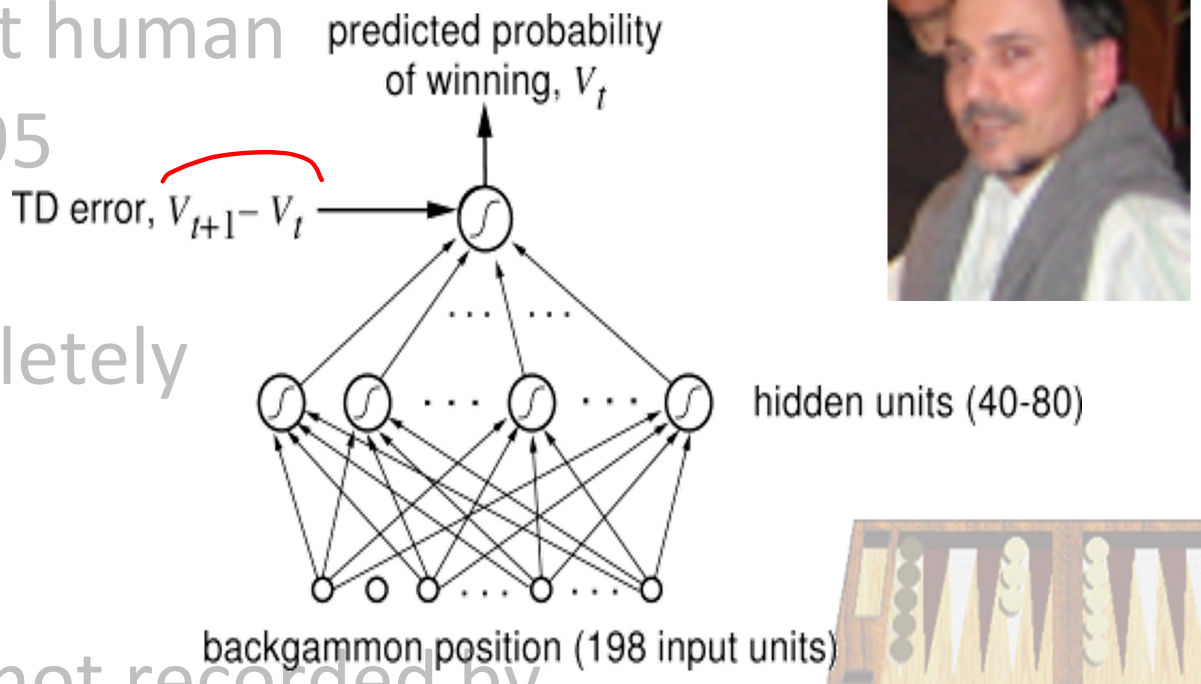
Human Level Backgammon player

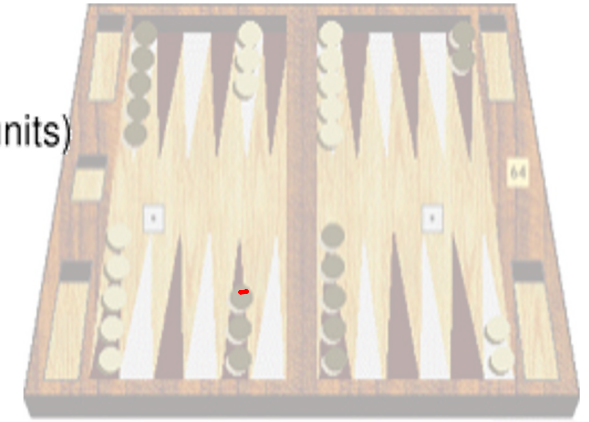
TD-Gammon (Tesauro 92, 94, 95)

Beat the best human player in 1995

Learnt completely by *self play*

New moves not recorded by humans in centuries of play





Human Level Backgammon player

TD-Gammon (Tesauro 92, 94, 95)

Beat the best human player in 1995

Learnt completely by *self play*

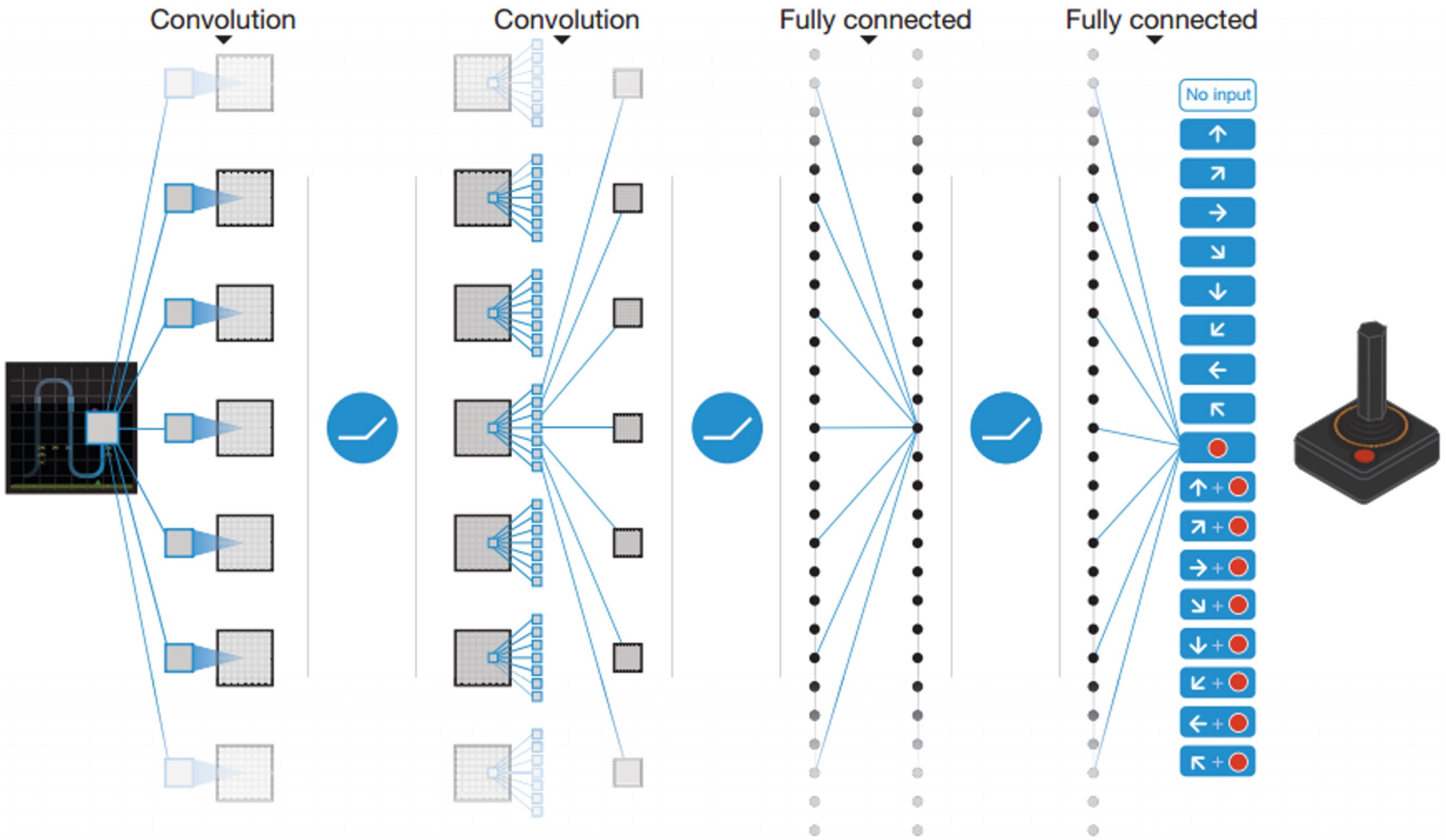
New moves not recorded by humans in centuries of play

What about the features?

18 actions

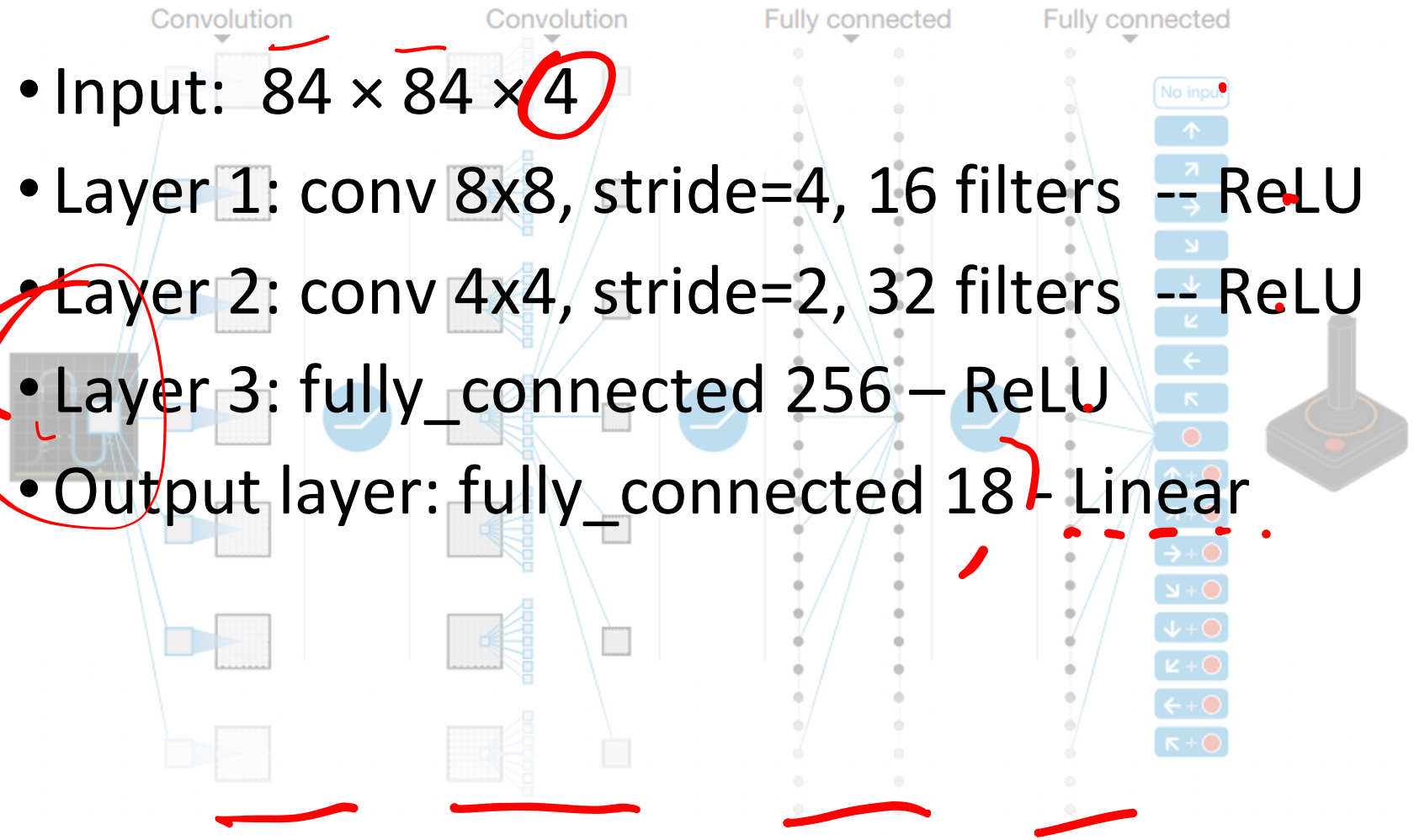
- Learnt to play from video input from **scratch!**

Deep Q-Learning Network (DQN)



Source: Deep Q Networks, Nature 2015

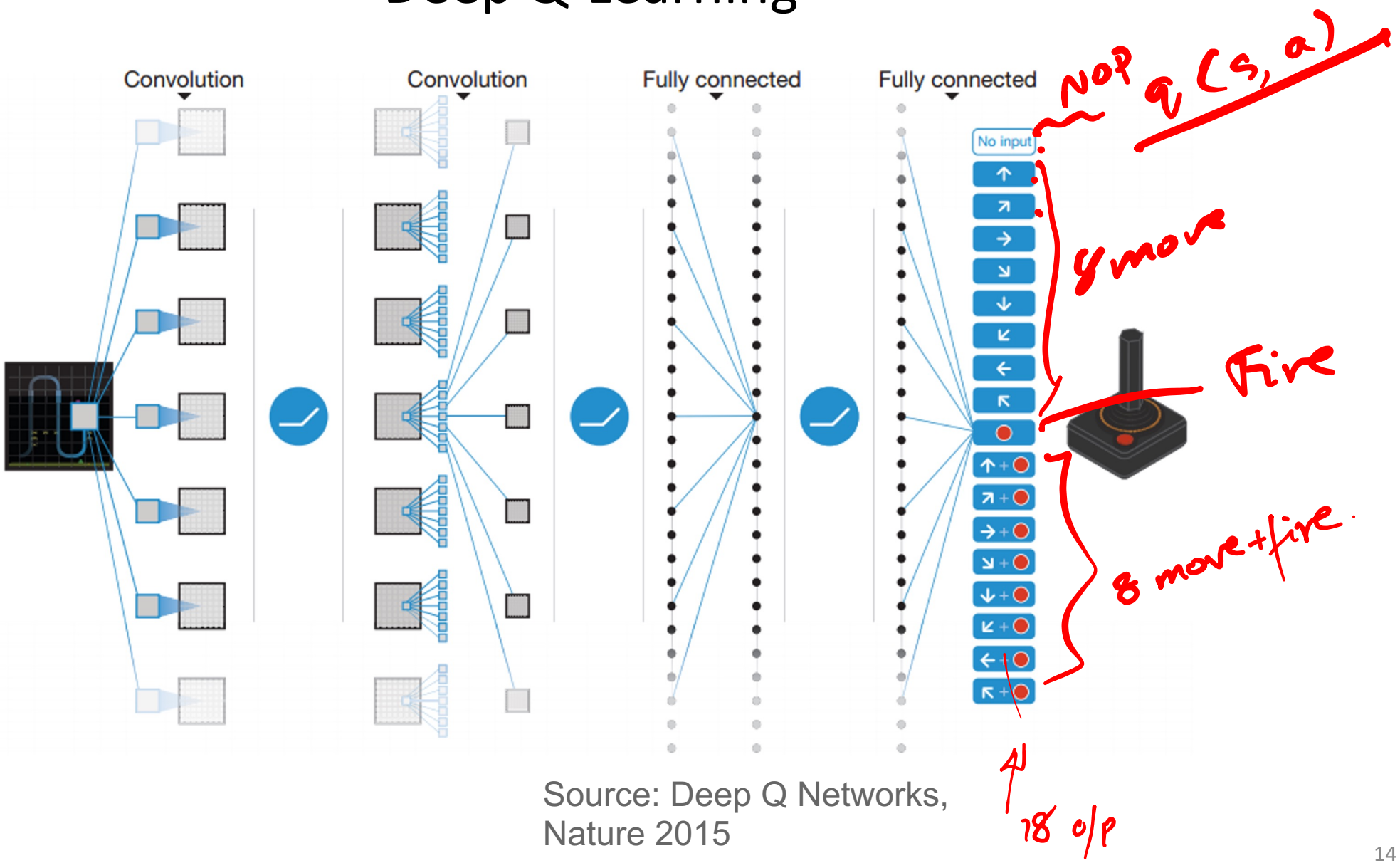
Deep Q-Learning



- Input: 84 × 84 × 4
- Layer 1: conv 8x8, stride=4, 16 filters -- ReLU
- Layer 2: conv 4x4, stride=2, 32 filters -- ReLU
- Layer 3: fully_connected 256 – ReLU
- Output layer: fully_connected 18 - Linear

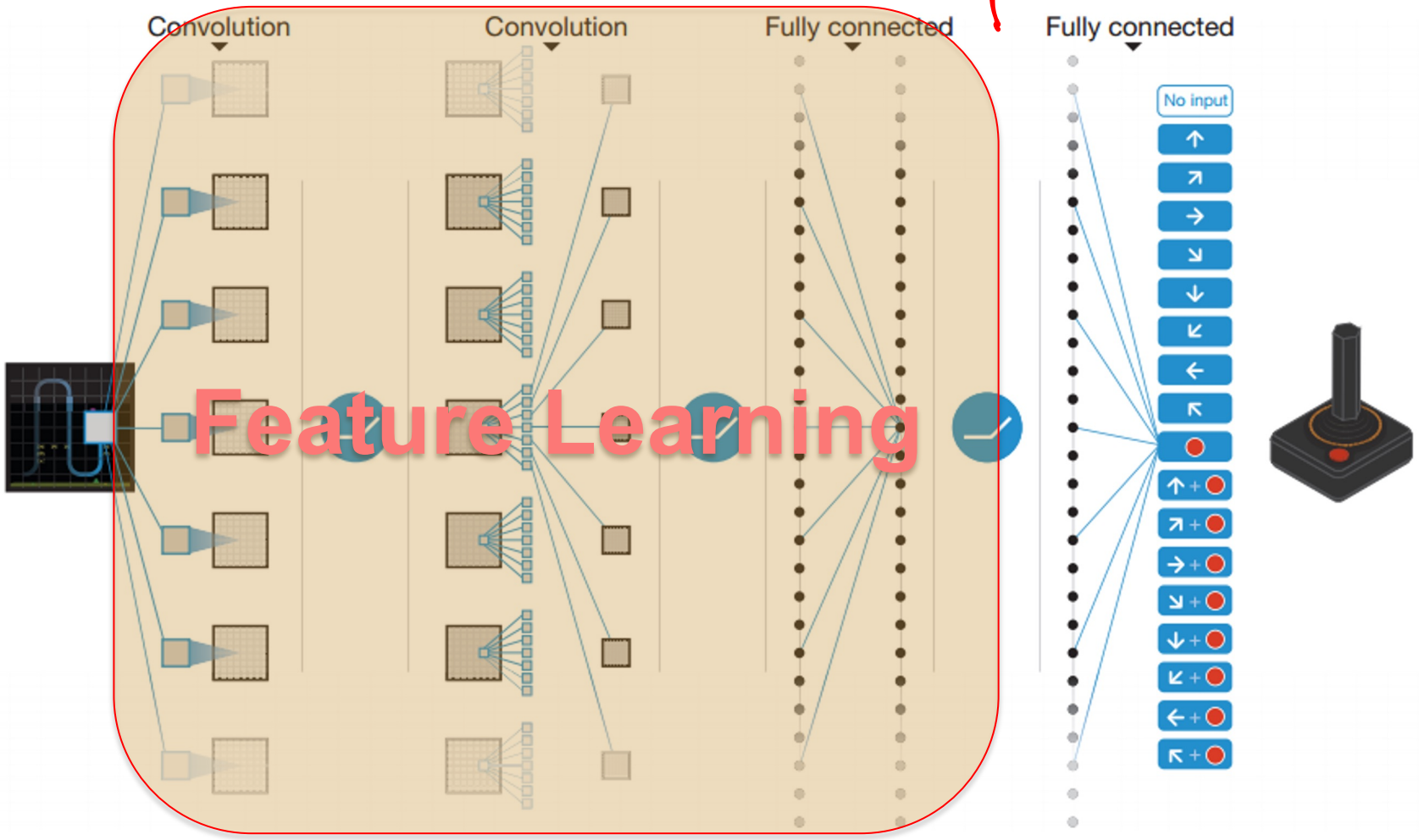
Source: Deep Q Networks, Nature 2015

Deep Q-Learning



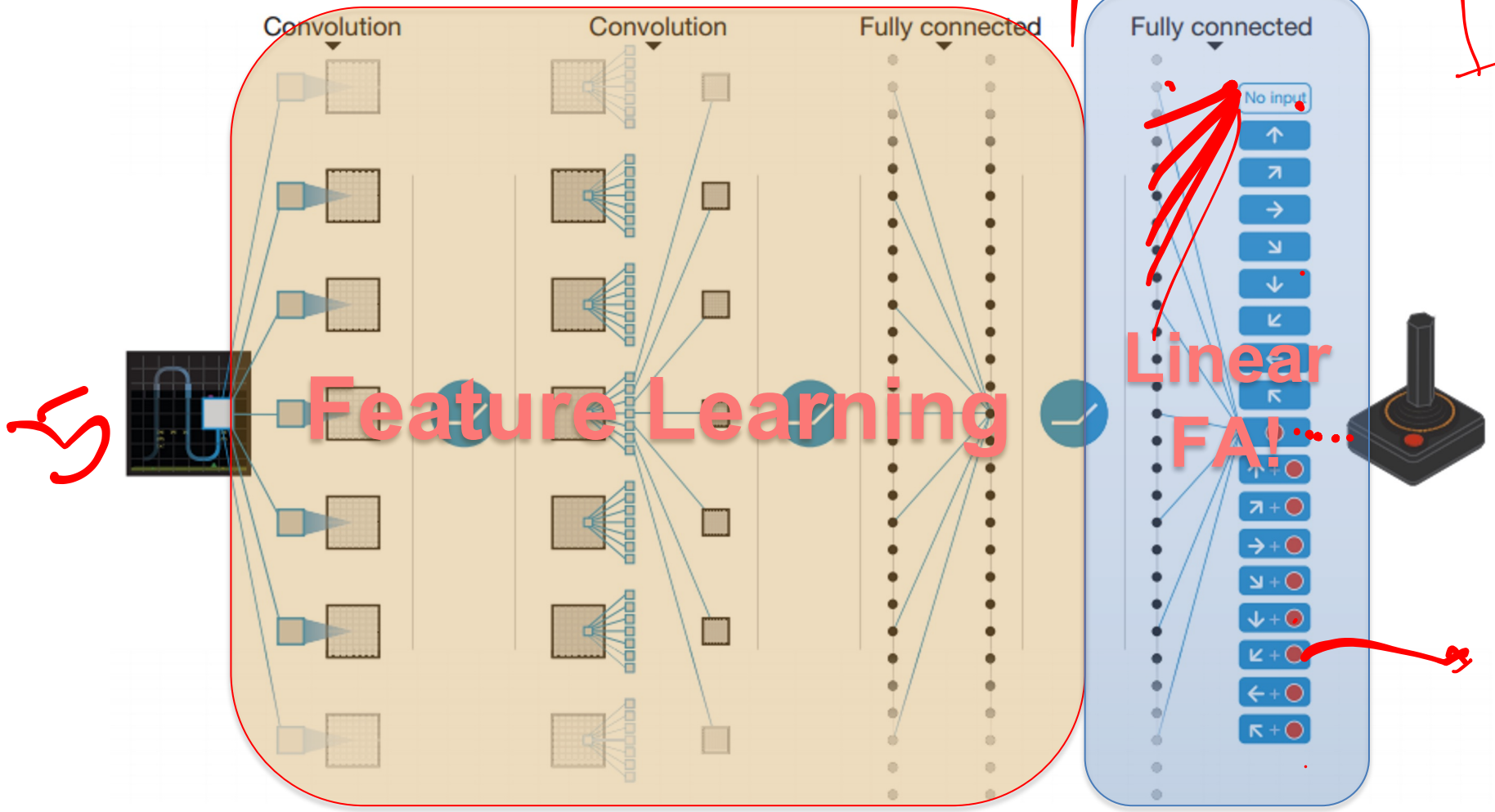
Source: Deep Q Networks, Nature 2015

Deep Q-Learning



Source: Deep Q Networks, Nature 2015

Deep Q-Learning



Source: Deep Q Networks, Nature 2015

Q-Network Learning

$$w_{t+1} = w_t - \frac{1}{2}\alpha \nabla_{w_t}\left[r_{t+1} + \gamma \max_a \hat{q}(s_{t+1}, a) - \hat{q}(s_t, a_t)\right]^2$$

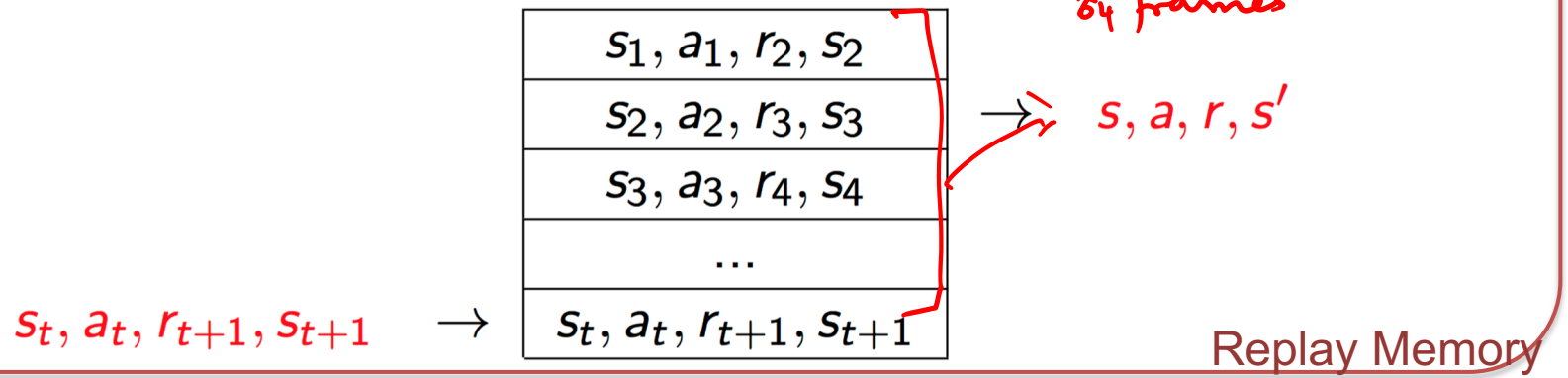
Divergence is an issue since the current network is used decide its own target

- Correlations between samples
- Non-stationarity of the targets

- How do we address these issues?
  - Replay Memory
  - Freeze target network

Q-Network Learning

To remove correlations, we build data-set from the agent's experience



Sample experiences from dataset; $\boldsymbol{w}^-$ frozen (with periodic updates) to address non-stationarity

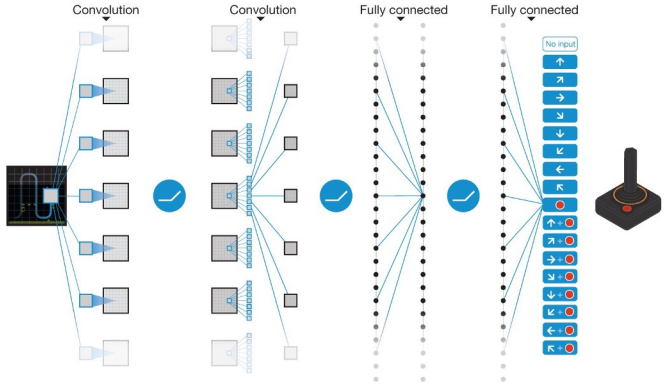
$$\left[\left\{r_{t+1} + \gamma \max_a \hat{q}\left(s_{t+1}, a; w^-\right)\right\} - \hat{q}\left(s_t, a_t; w\right)\right]^2$$

Freeze Target Network

Deep-Q Networks

Architecture:-

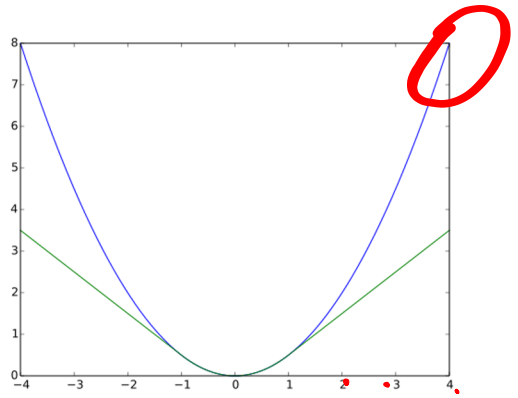
1. Has a set of convolutional layers which act as feature extractors.
2. These features are then passed through a series of fully connected layers.
3. The output layer has |A| number of nodes which are used to calculate the Q-value for each action.

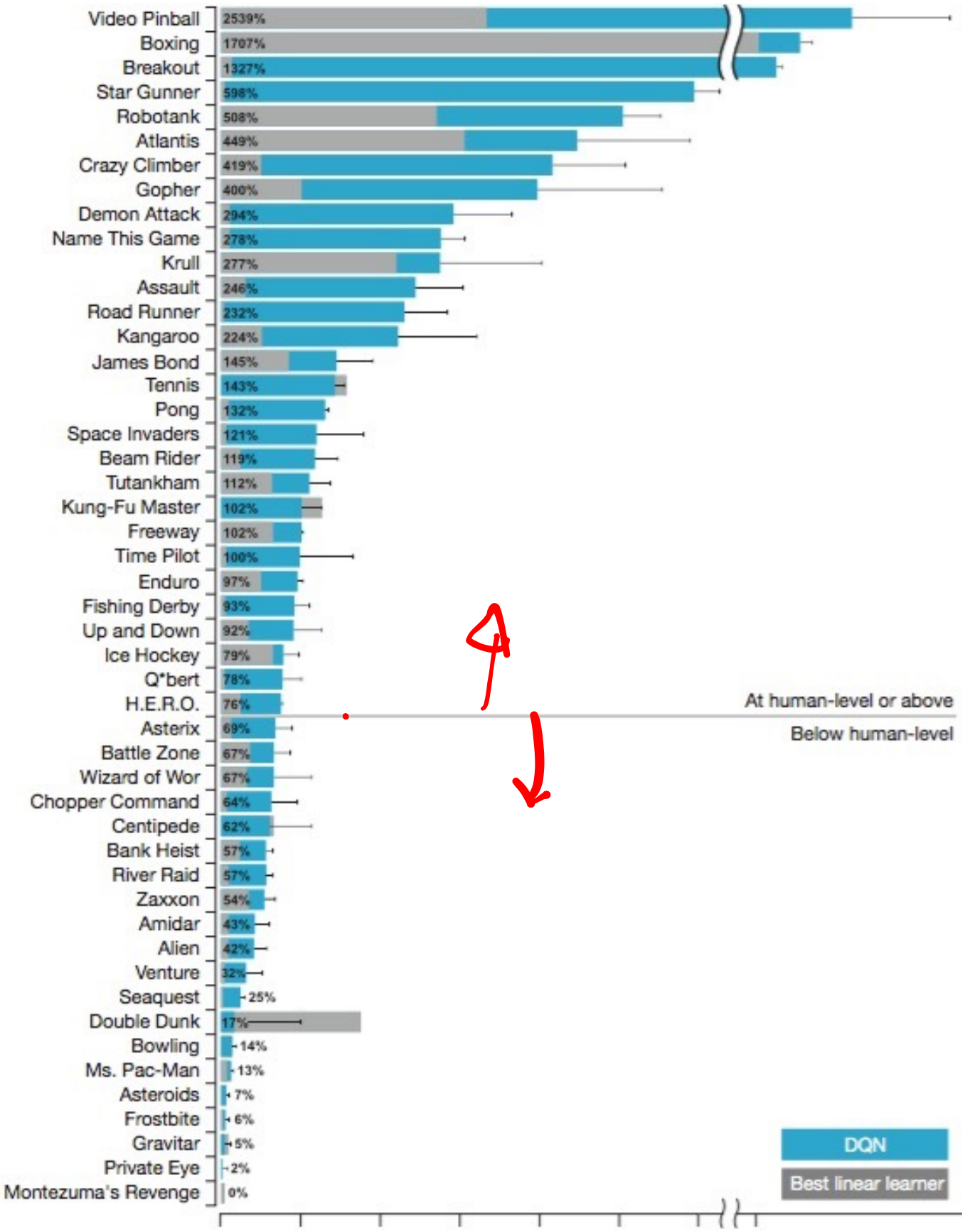


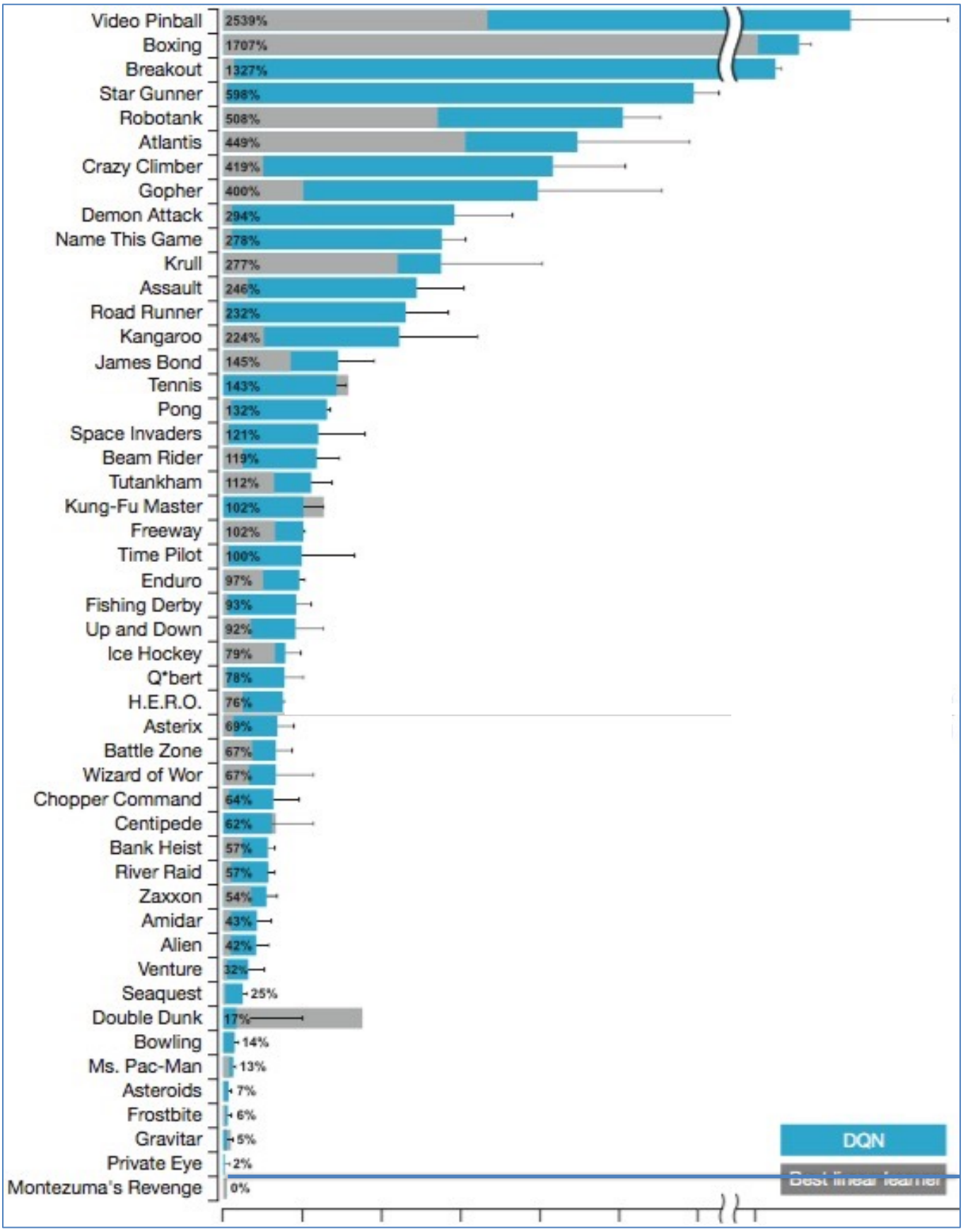
The above network is updated using huber loss and not regular least squares loss.

Below is the regular expression for huber loss.

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \le \delta, \\ \delta(|a| - \frac{1}{2}\delta), & \text{otherwise.} \end{cases}$$




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%
At human-level or above
Below human-level
DQN
Best linear learner


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%
DQN