COMMONWEALTH OF AUSTRALIA

Copyright Regulations 1969

WARNING

This material has been reproduced and communicated to you by or on behalf of Monash University pursuant to Part VB of the *Copyright Act 1968* (the Act). The material in this communication may be subject to copyright under the Act. Any further reproduction or communication of this material by you may be the subject of copyright protection under the Act. Do not remove this notice.

Acknowledgements

This material includes content adapted from instructional resources made available by David Silver as part of his Reinforcement Learning course at UCL under CC-BY-NC 4.0.

Refer to https://www.davidsilver.uk/teaching/ for full details.

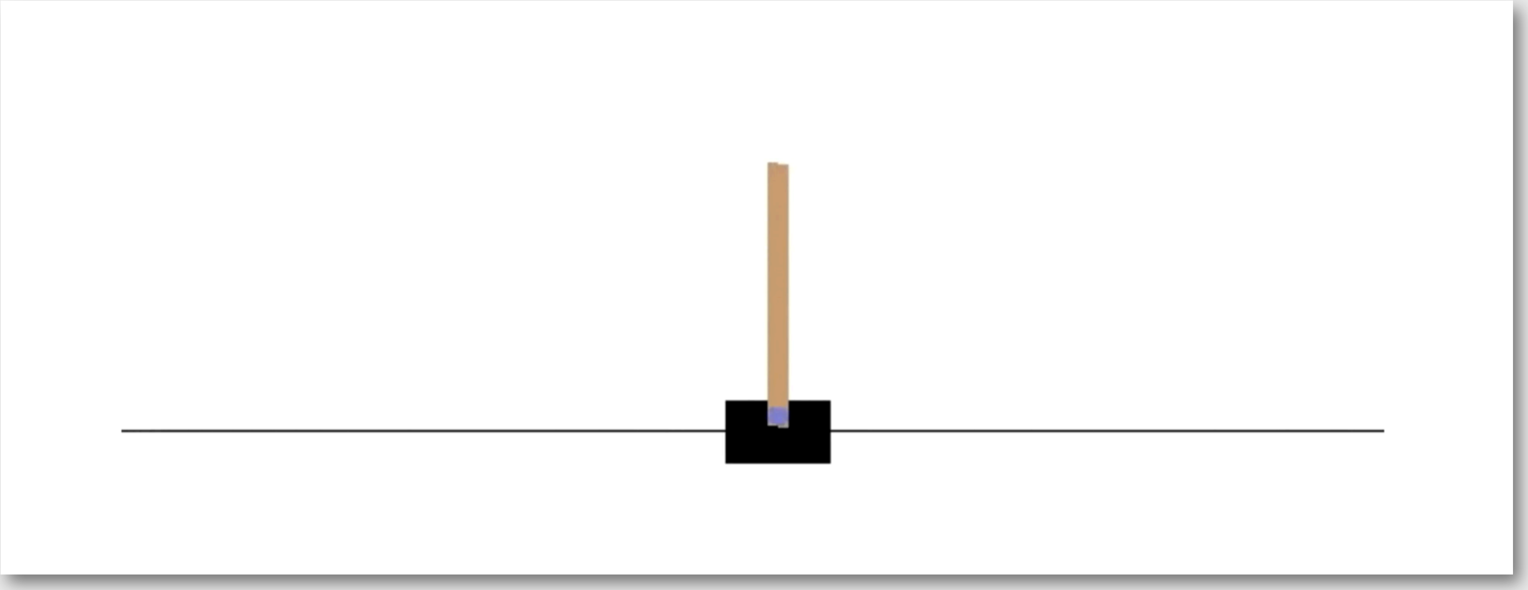
MONASH University
Information Technology

FIT5226
Multi-agent System & Collective Behaviour

Wk 5: Deep Reinforcement and Deep-Q

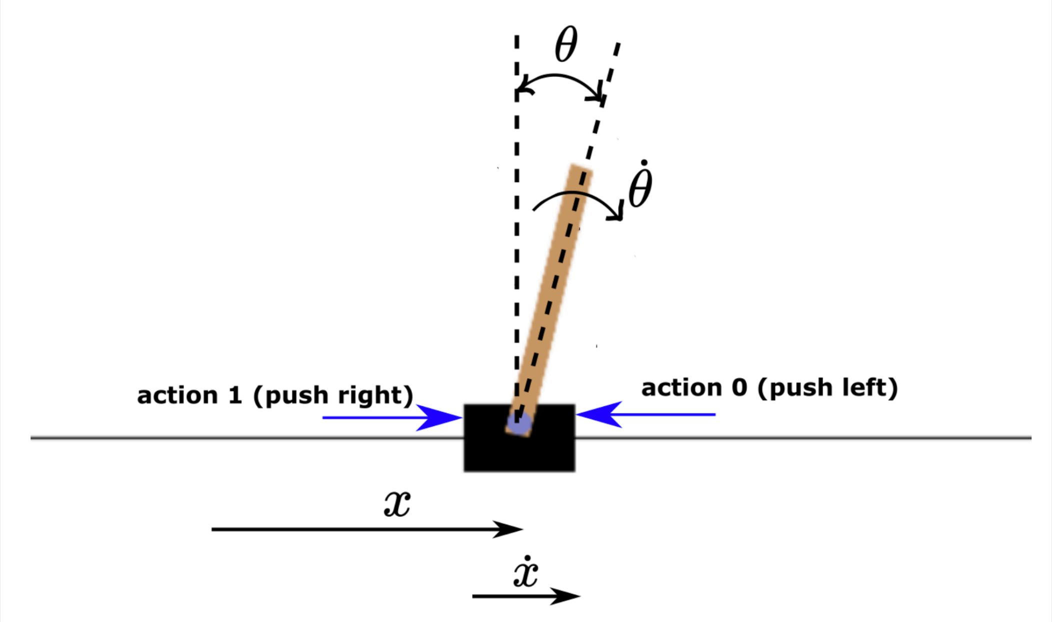
Bernd Meyer, August 2024

The Cart Pole Problem



cart pole *or* inverted pendulum

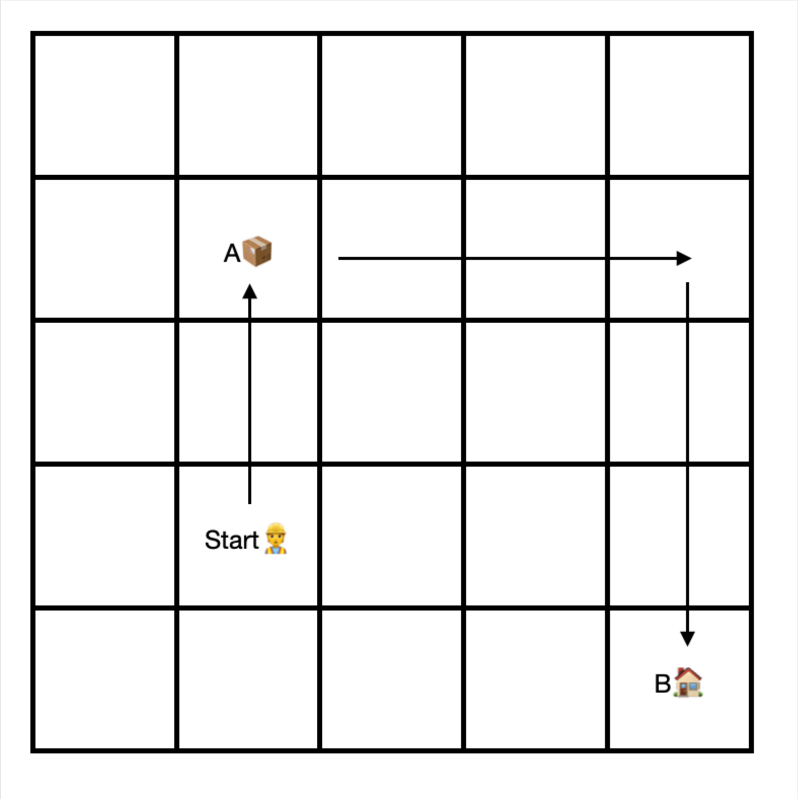
Continuous State Spaces

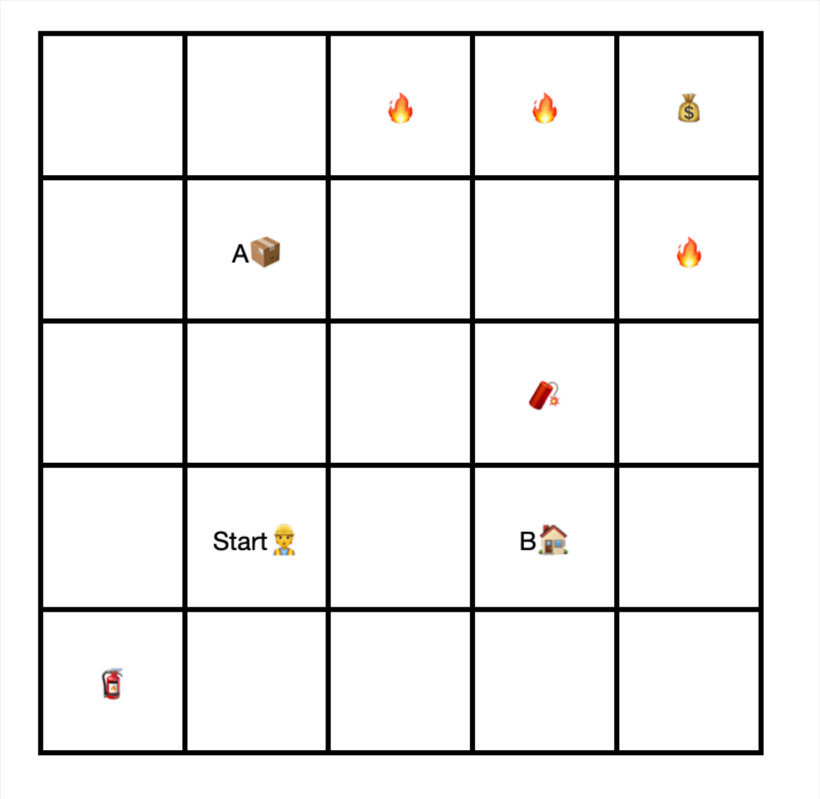


Observation (state) space

- Cart Position (x): The horizontal position of the cart on the track.
- Cart Velocity (dx/dt): The velocity of the cart.
- Pole Angle (θ): The angle of the pole from the vertical upright position.
- Pole Angular Velocity (dθ/dt): The angular velocity of the pole.

State Space Size





$\approx 25^2 = 625$

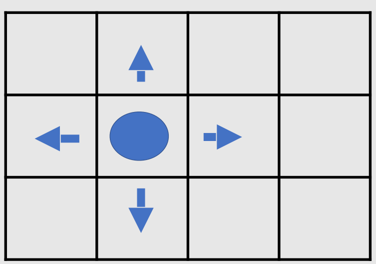
$\approx 25^9 = 3{,}814{,}697{,}265{,}625$

Backgammon: 10^{20}; Go: 10^{170}

Generalisation in more complex state spaces

- Mars rover must learn high-level control to traverse terrain
- Driving over hazardous terrain 10x slower

4 actions



Distance sensor

4	3	2	1
3	2	1	G
S	3	2	1

Terrain sensor

-1	-1	-10	-10
-1	-1	-1	-1
-1	-10	-1	-10

State Space for Mars Rover

- State space is a vector of features

$$s = \left(D_N, D_E, D_S, D_W,\ H_N, H_E, H_s, H_W\right)$$
$$s = (3, 2, 3, 4, 0, 1, 0, 0)$$

- Suppose:
  - Max distance 5, Boolean hazard sensor,
  - Then, $\left|S\right| = 5^4 + 2^4 = 10{,}000$ states, $\left|A\right| = 4$ actions,
  - 40,000 Q-table entries (about 157 kB of memory)

Generalisation

- After some exploration, our rover learned to navigate safely

- Now we change the map:

-1	-1	-10	-10
-1	-1	-1	G
S	-10	-1	-10

→

-1	-1	-1	-10
-1	-10	-1	G
S	-10	-1	-10

- What will happen?

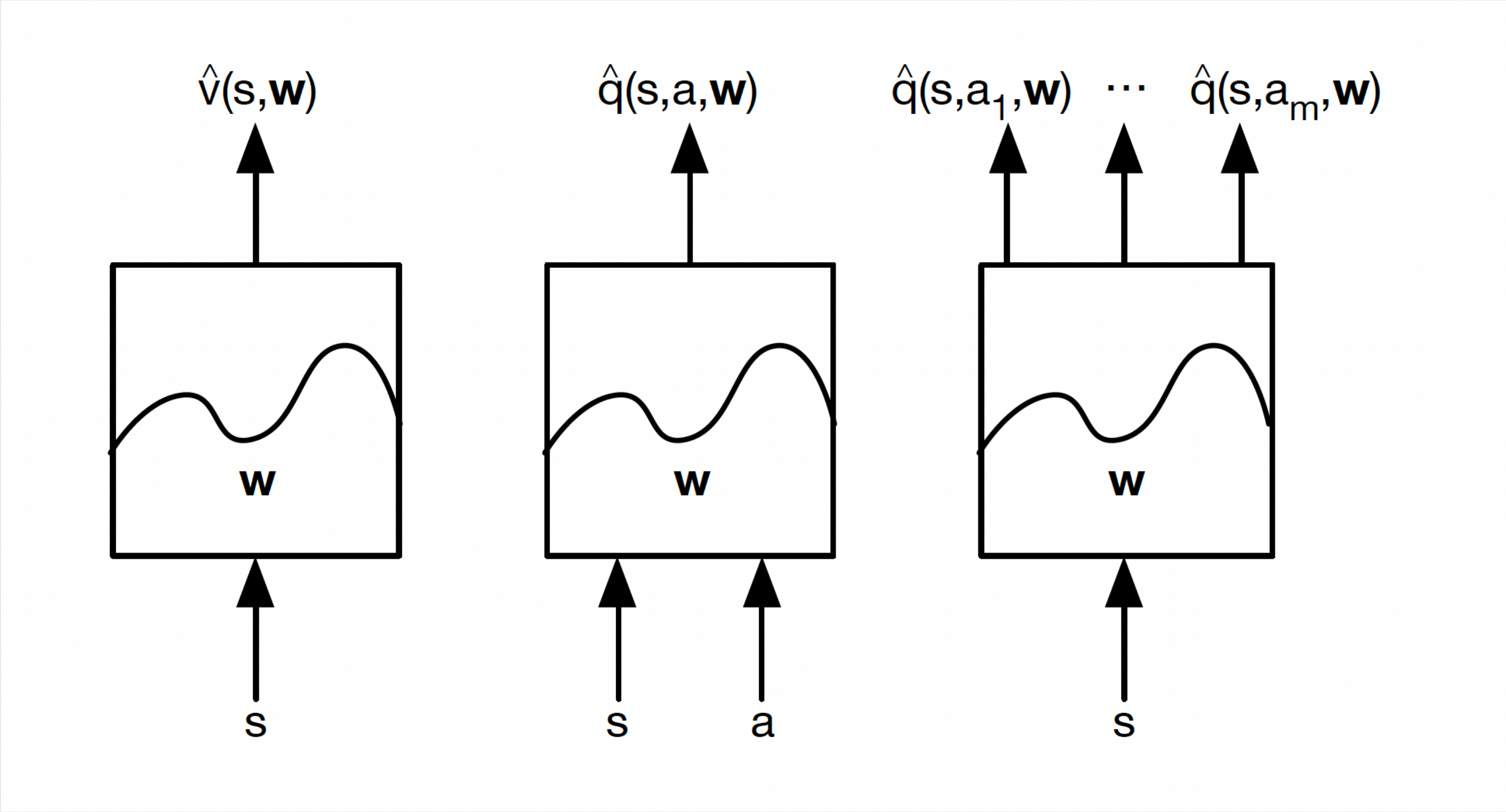
Value Function Approximation

- Very large state spaces
- Continuous state spaces
- Need for generalisation

➡ use a value function instead of a value table

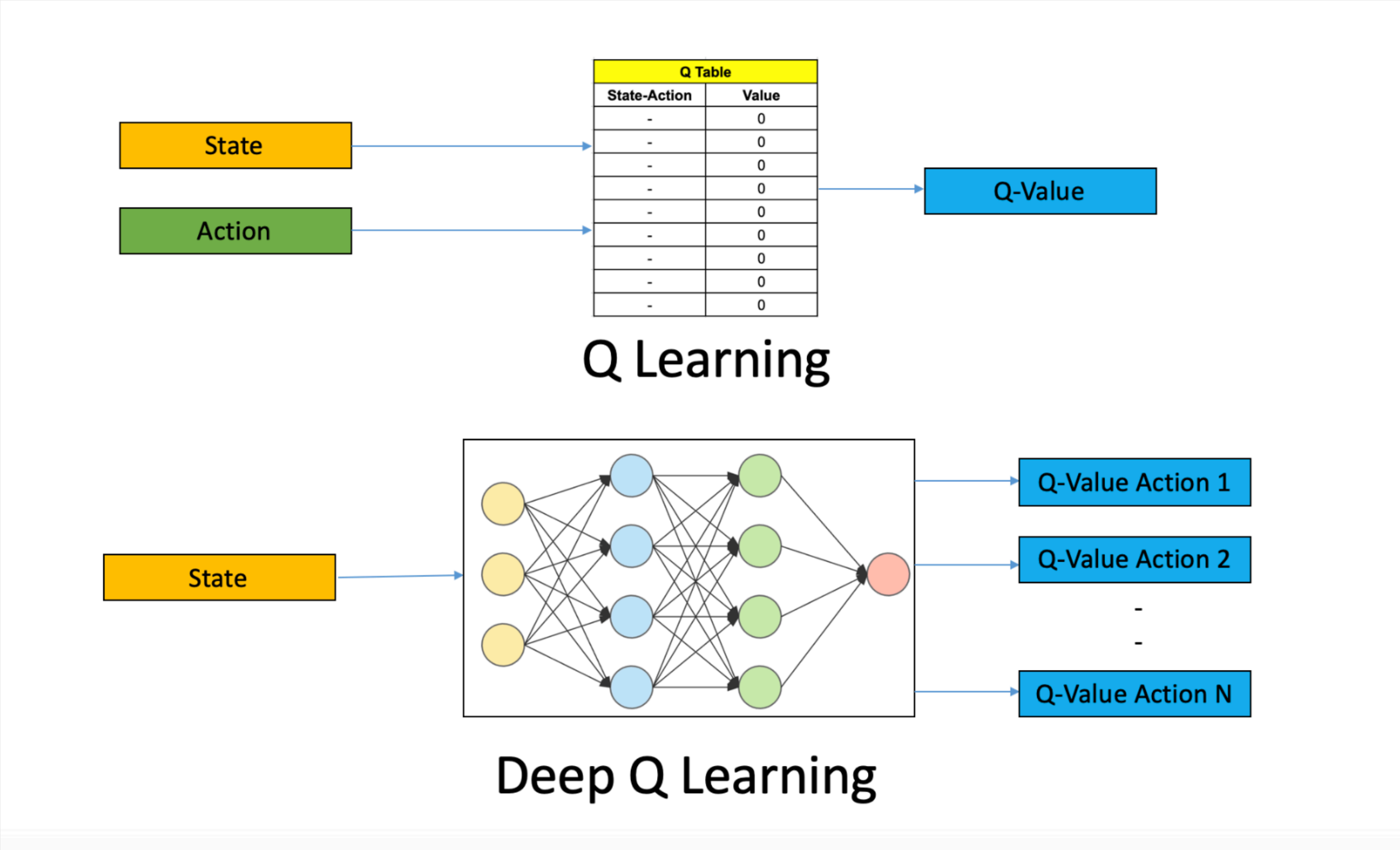
of course, this can only be approximate

Value Function Approximation Types



$F : S \rightarrow \mathbb{R}$ $F : S \times A \rightarrow \mathbb{R}$ $F : S \rightarrow A \times \mathbb{R}$

Value Function Approximation with Deep-Q Network

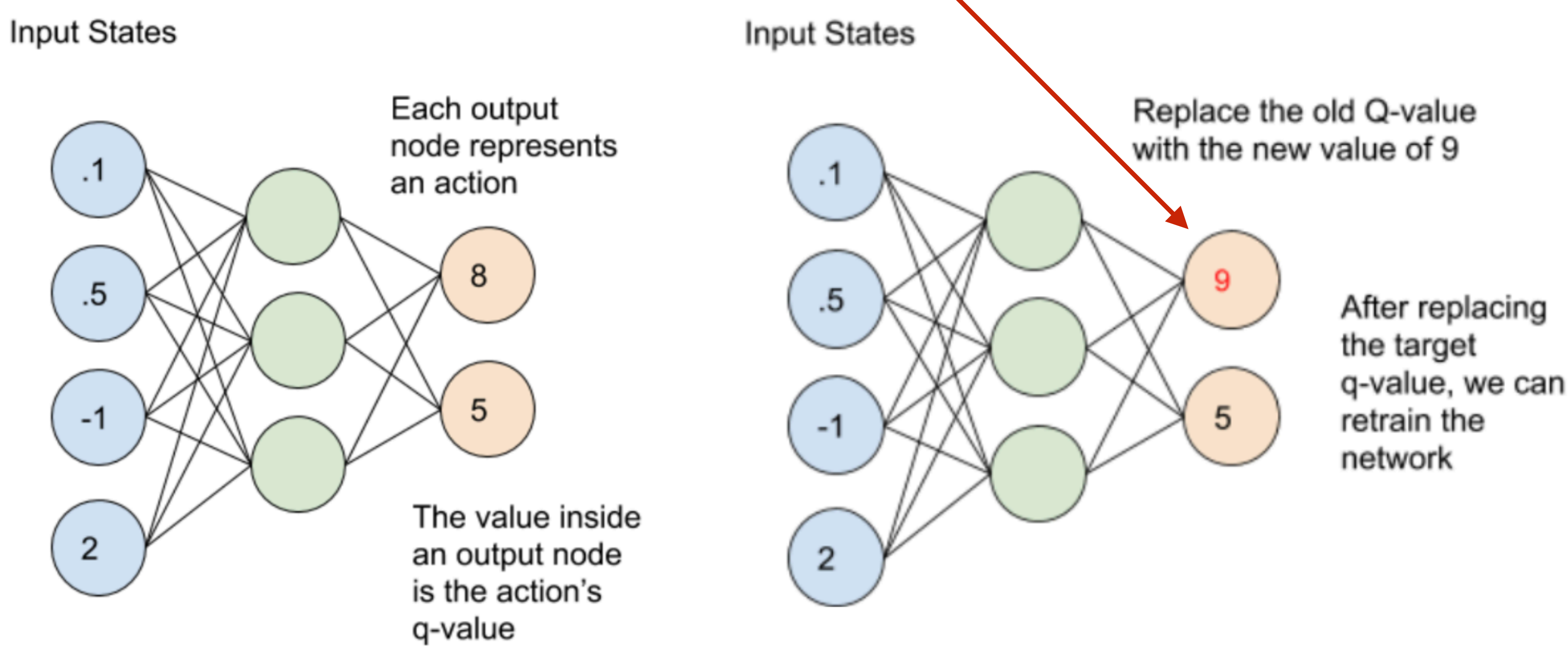


from: https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/

From Bellman Update to Loss Function

$$Q(S_t, A_t) = (1 - \alpha)\, Q(S_t, A_t) + \alpha * (R_t + \lambda * max_a\, Q(S_{t+1}, a))$$

TD-target



DQN Algorithm

```
Initialize the Agent to interact with the Environment
while not converged do
    /* Sample phase
    ϵ ← setting new epsilon with ϵ-decay
    Choose an action a from state s using policy ϵ-greedy(Q)
    Agent takes action a, observe reward r, and next state s'
    Store transition (s, a, r, s', done) in the experience replay memory D

    if enough experiences in D then
        /* Learn phase
        Sample a random minibatch of N transitions from D
        for every transition (s_i, a_i, r_i, s'_i, done_i) in minibatch do
            if done_i then
                y_i = r_i
            else
                y_i = r_i + γ max_{a'∈A} Q̂(s'_i, a')
            end
        end
        Calculate the loss L = 1/N Σ_{i=0}^{N-1} (Q(s_i, a_i) − y_i)^2
        Update Q using the SGD algorithm by minimizing the loss L
        Every C steps, copy weights from Q to Q̂
    end
end
```

Experience Replay

- Equivalent to standard Q-learning: adapt the network for every action as it is performed (usually using a single SDG step and MSE loss)

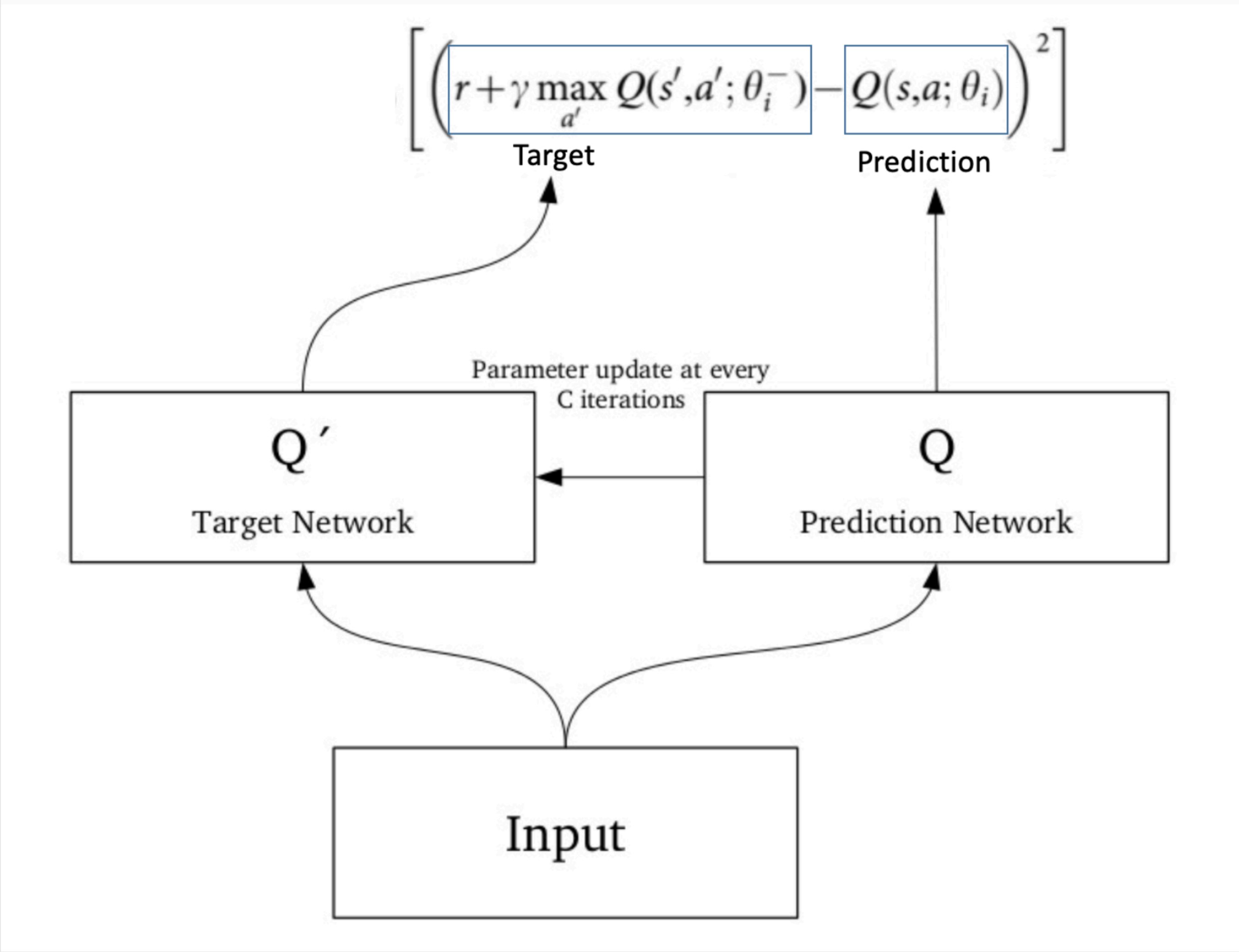
- **This can be unstable. To stabilise we perform *Experience Replay***

- All Transitions (state, action, reward, new state) are stored in a ring buffer of limited size

- Instead of just using the least action for each training step we use a mini-batch to update
    - randomly sample a mini-batch from the buffer
    - compute loss and back-propagate for the whole mini-batch
    - we may do this only every *k* actions

- Note that this if <u>off-policy</u>
  (and thus requires an off-policy base algorithm like Q-learning)

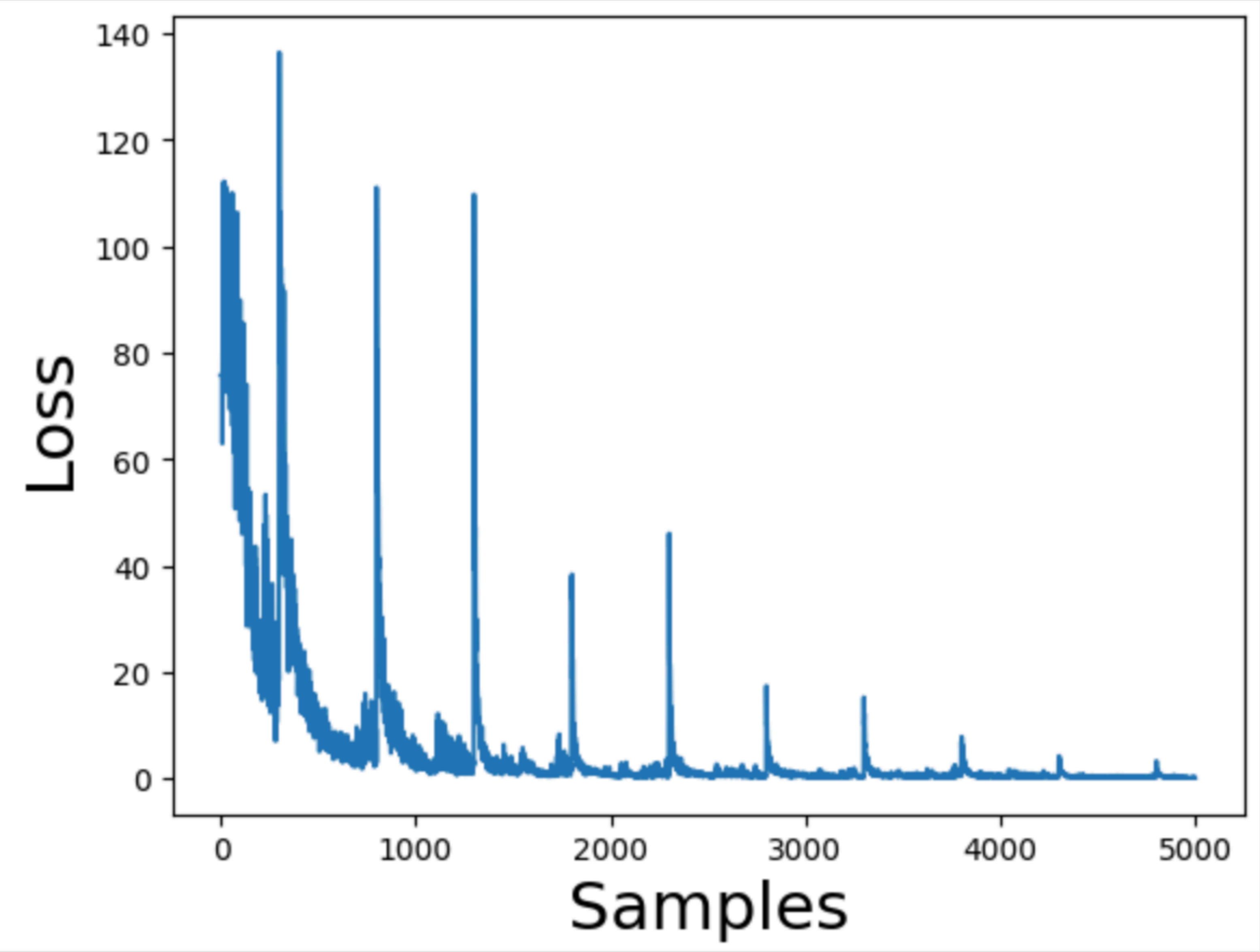
Target Network versus Prediction (main) network



$= \mathscr{L}$

from: https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/

Updating the Target Network

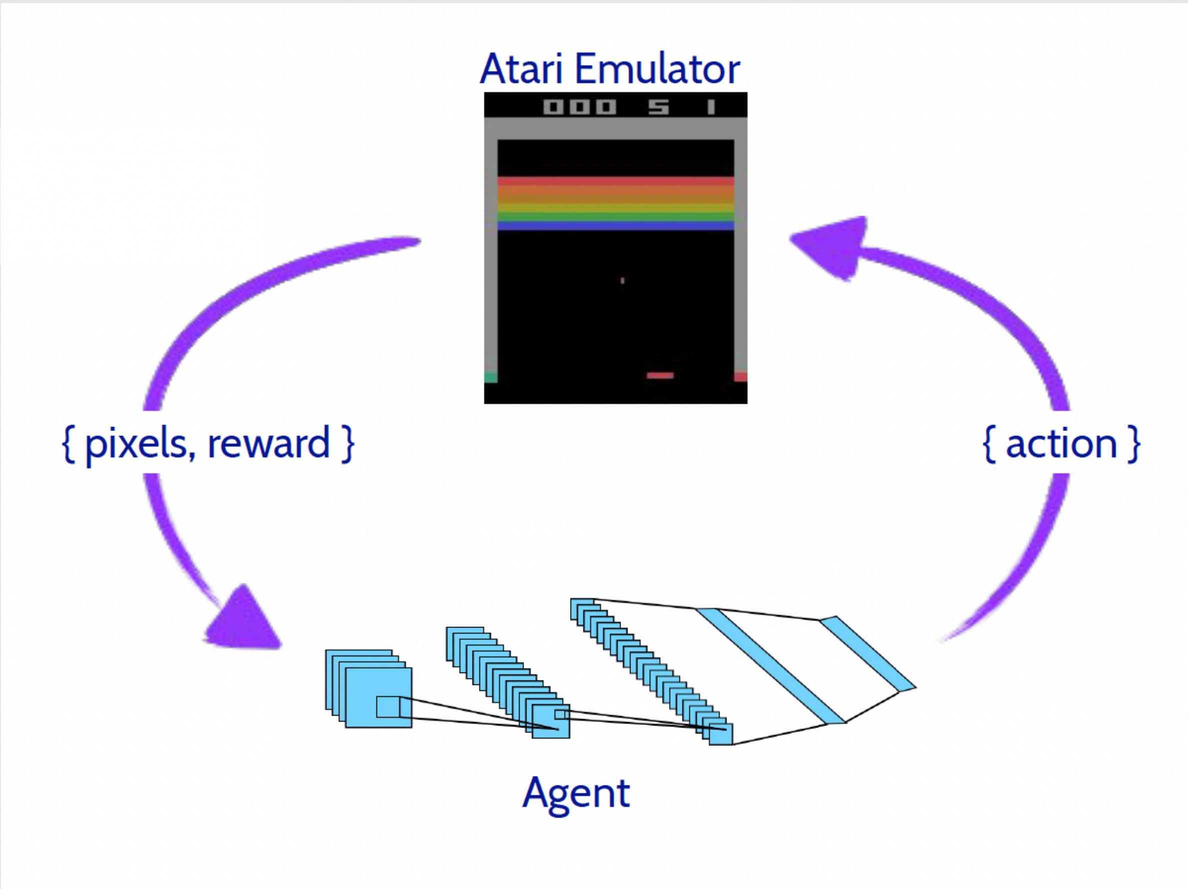


Noisy-fying the state

- Discrete states (e.g. in grid wolds), especially one-hot can lead to “Dead Neurons”
- A trick to avoid this is to add a small amount of noise to the states
- This problem occurs specifically with ReLU (rectified linear unit) activation functions since these are non-differentiable at zero when the state vector is sparse (i.e. most its elements are zero) as is the case, for example, in our grid worlds in the lab.

```
state = state.reshape(1,k) + np.random.rand(1,k)/c
```

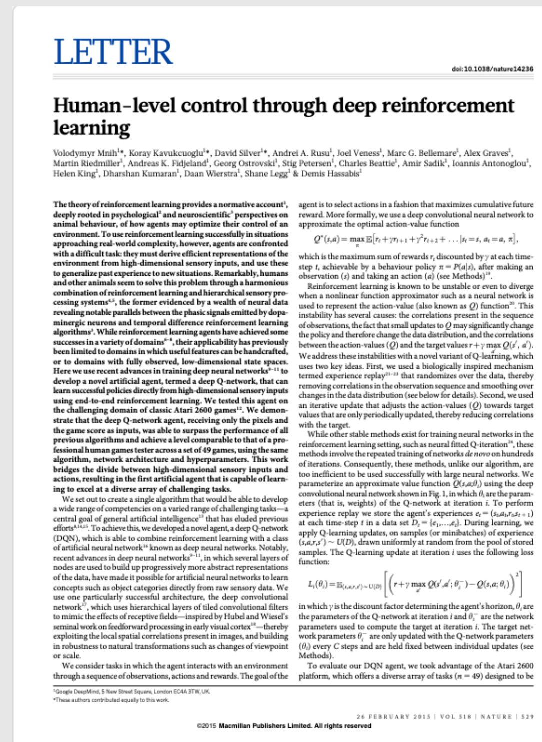
End-to-end State Spaces



Nature Journal
https://www.nature.com › letters

Human-level control through deep reinforcement learning

by V Mnih · 2015 · Cited by 25488 — The theory of **reinforcement** learning provides a normative account, **deeply** rooted in psychological and neuroscientific perspectives on animal ...



LETTER

doi:10.1038/nature14236

Human-level control through deep reinforcement learning

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]

Breakout (DeepMind)

Image Feature Vectors

RED	GREEN	BLUE
0	0	0
255	255	255
0	0	0
210	210	0
0	0	0
210	210	0
0	0	0

State space for Atari Games

- We could use the image pixel values directly,
  - Vector $(p^R_{1,1}, p^G_{1,1}, p^B_{1,1}, p^R_{1,2}, \ldots, p^R_{h,w}, p^G_{h,w}, p^B_{h,w})$
  - Atari frame, 3 x 210 x 160 = 100,800 inputs
- Two problems:
  - Huge input size;
  - Is a single frame enough for decisions?

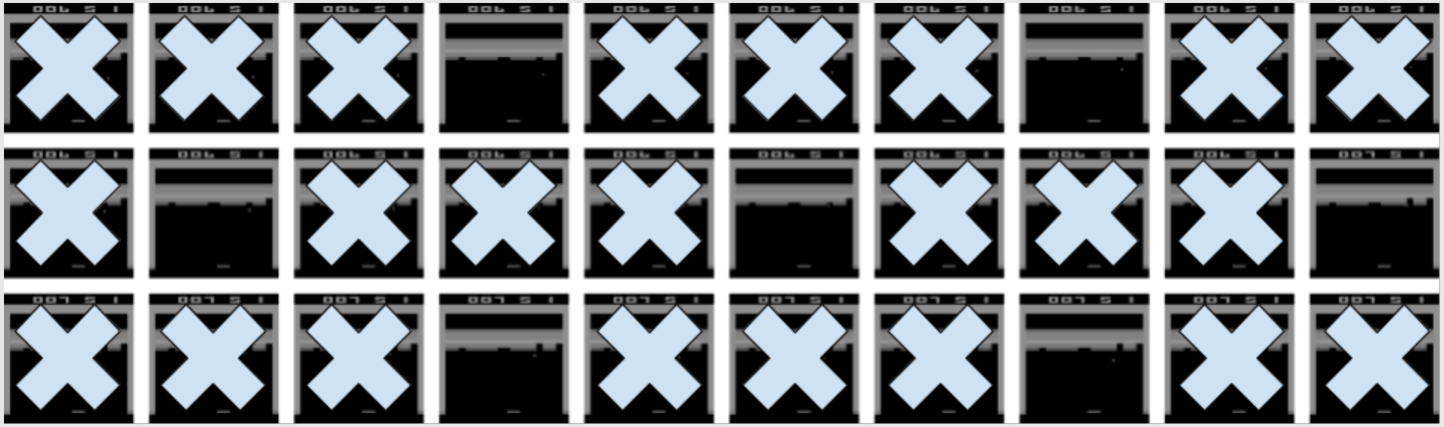
Movement

Which direction is the ball moving?

Encoding Motion

- Accurate control needs motion information,
  - Use multiple frames
- Reduce size of the input image by
  - Using greyscale,
  - Rescaling frame size to 84 x 84

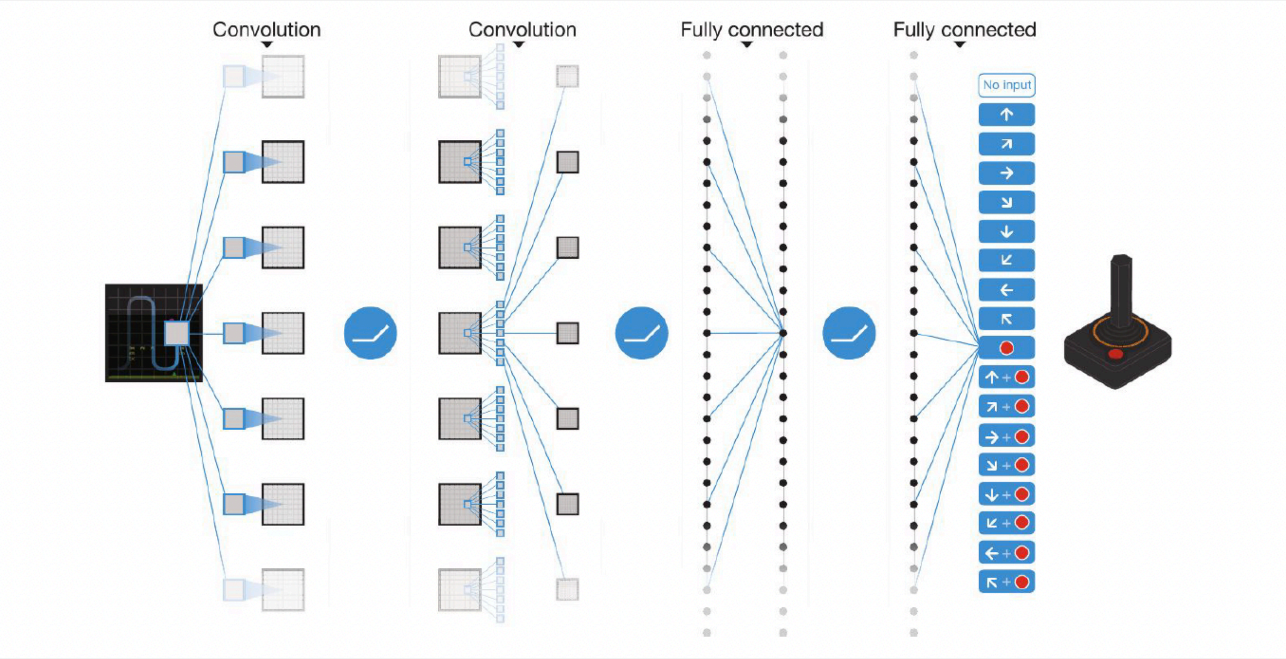
Subsampling



2-dimensional input (spatial context)

- By reducing input size we end up with 4 x 84 x 84 = 28,224 pixels
- Flattening these into an input layer is possible, **but loses spatial context**
- Solution: use Convolutional Neural Networks

Q-Network Structure



- s = 4 x 84 x 84 grayscale input,
- 32 filters of 8x8 with stride 4,
- 64 filters of 4x4 with stride 2,
- 64 filters of 3x3 with stride 1,
- Fully connected, 512 units
- Fully connected, |A|=18 output

Advanced Deep-RL Methods

DQN (2013)

Double DQN

Duelling DQN

Prioritized Replay

Multi-step returns

Distributional Returns (C51)

Rainbow (2018)

Implicit Quantile Nets (IQN)

Munchausen RL (M-IQN; 2020)

And many more algorithms from other base algorithms, like deep policy gradients, actor critics

Take home lessons

- Large state spaces, continuous state spaces and generalisation require function approximation
- We can use neural networks to approximate the state(action) value function
- Deep-Q = Q-Learning + (DNN substitutes Q-Table)
    - the loss is calculated as difference to the TD-target
- Experience replay is required to improve learning
- Dual networks are required to avoid catastrophic forgetting
- Adding Noise helps to avoid “Dead neurons”
- Deep-Q is only just the beginning …