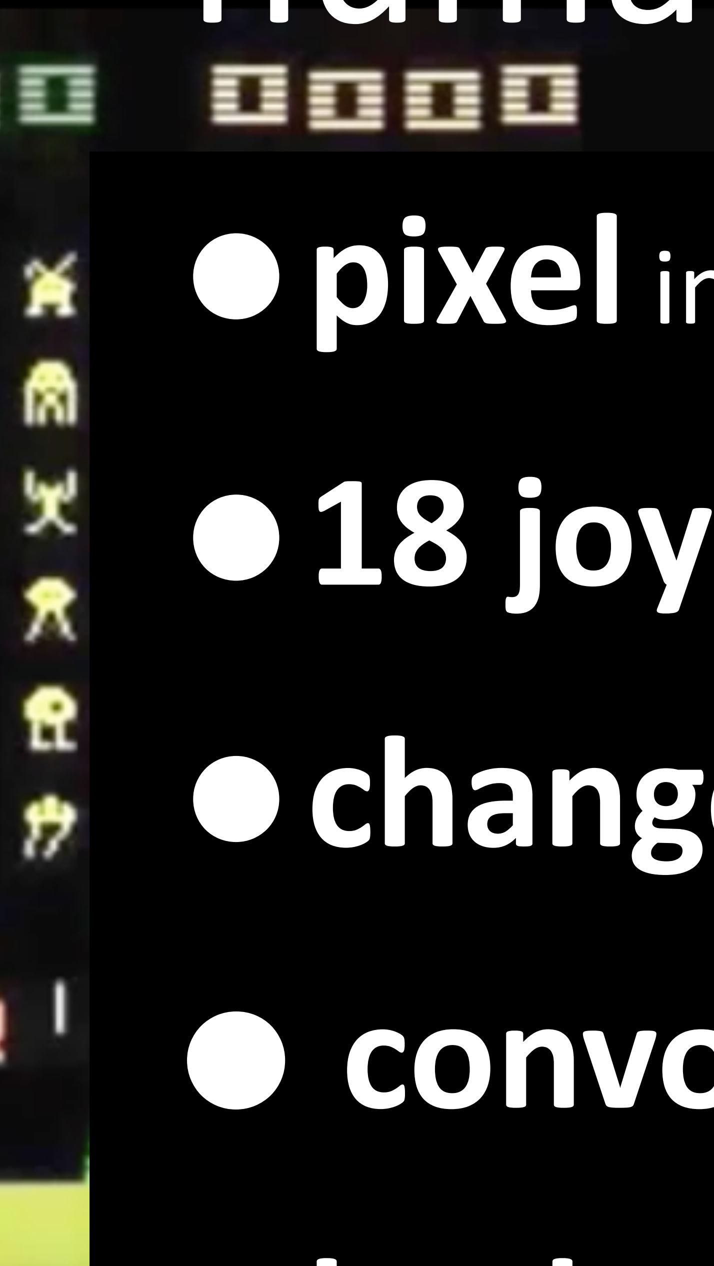
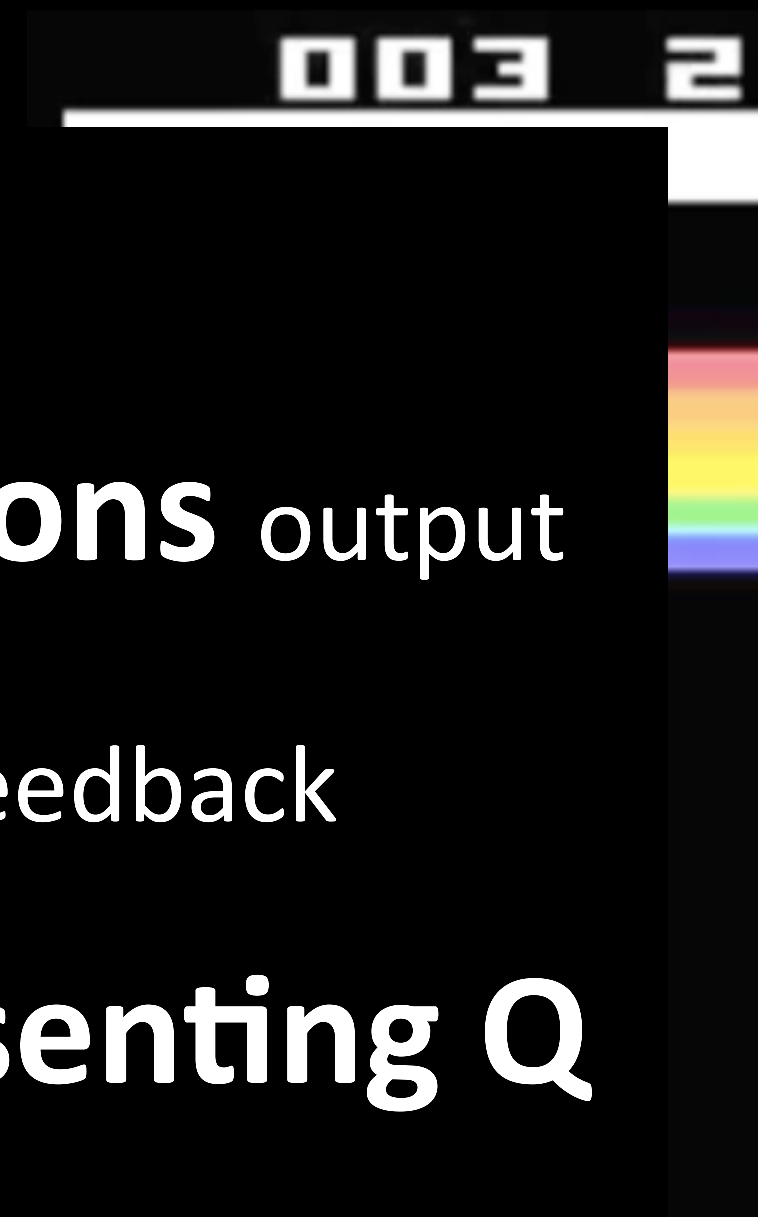
scaling up RL with function approximation

human level game control

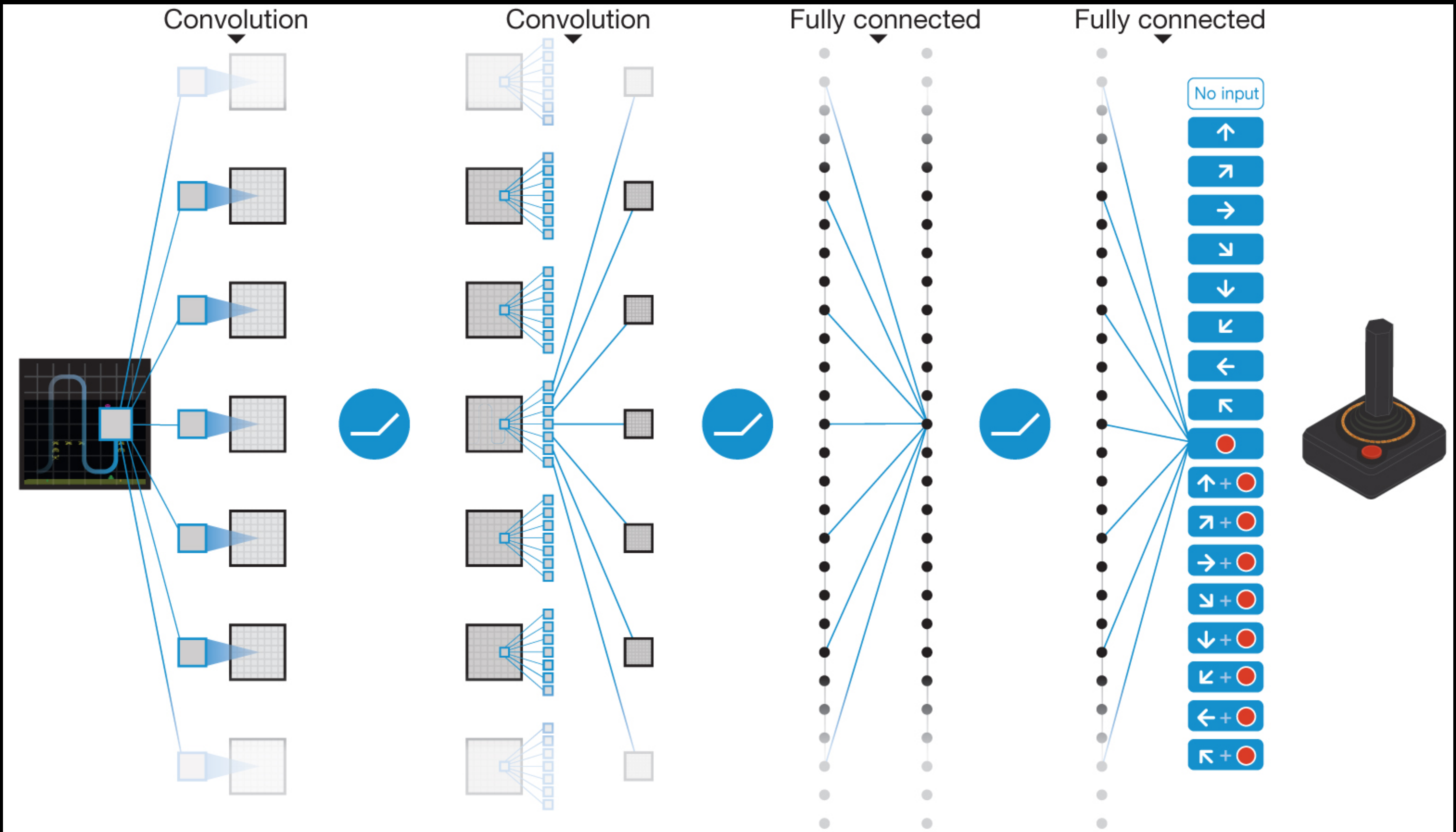




- **pixel** input
- **18 joystick/button positions** output
- **change in game score** as feedback
- **convolutional net representing Q**
- **backpropagation** for training!

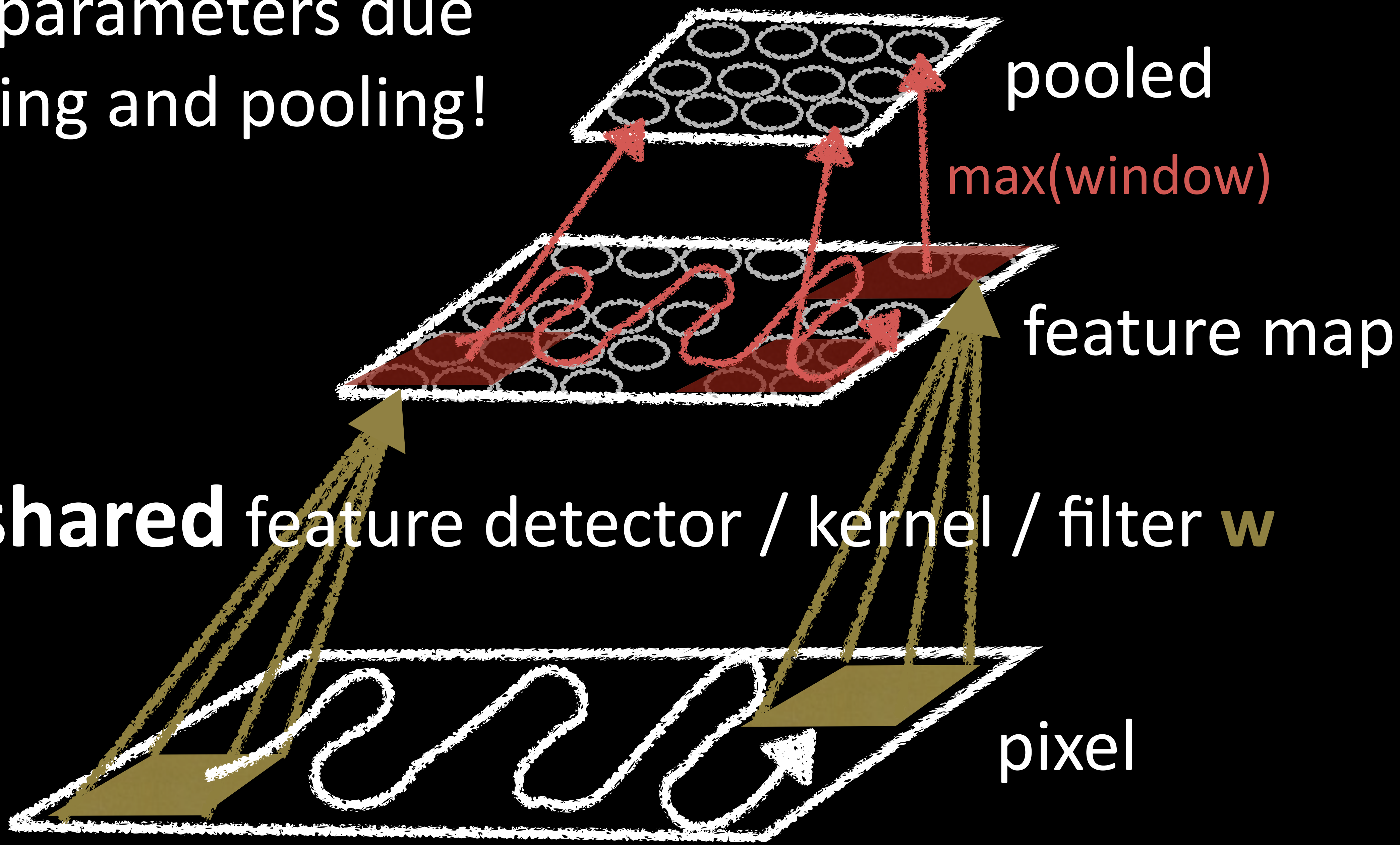
Human-level control through deep reinforcement learning,
Mnih et. al., Nature 518, Feb 2015
http://www.nature.com/nature/journal/v518/n7540/full/nature14236.html

neural network

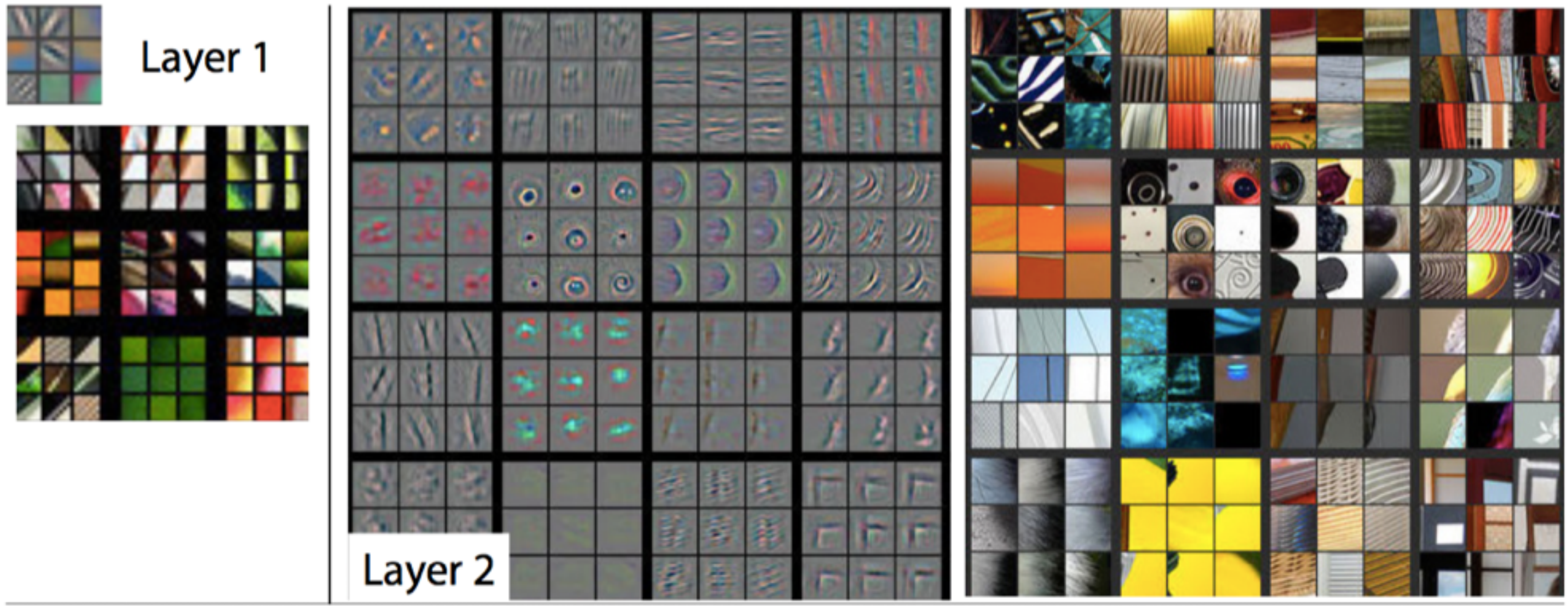


convolution, weight sharing, and pooling

fewer parameters due to sharing and pooling!

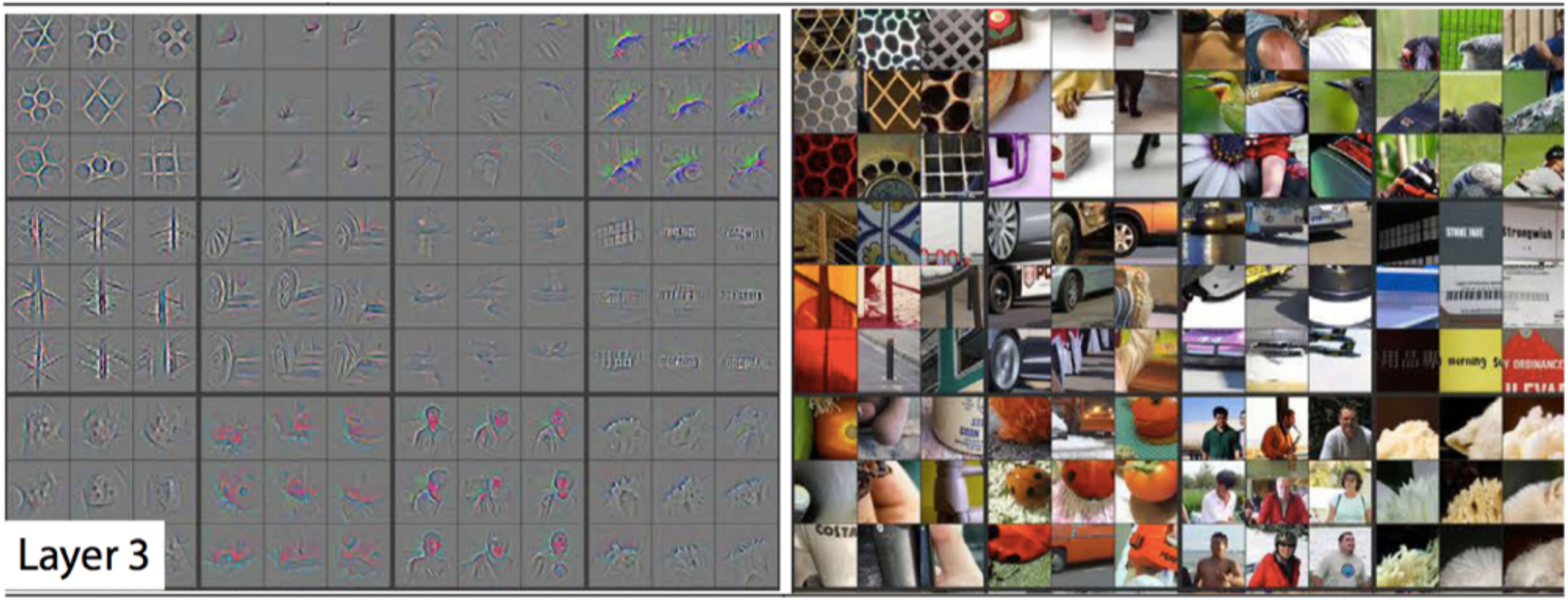


what does a deep neural network do?

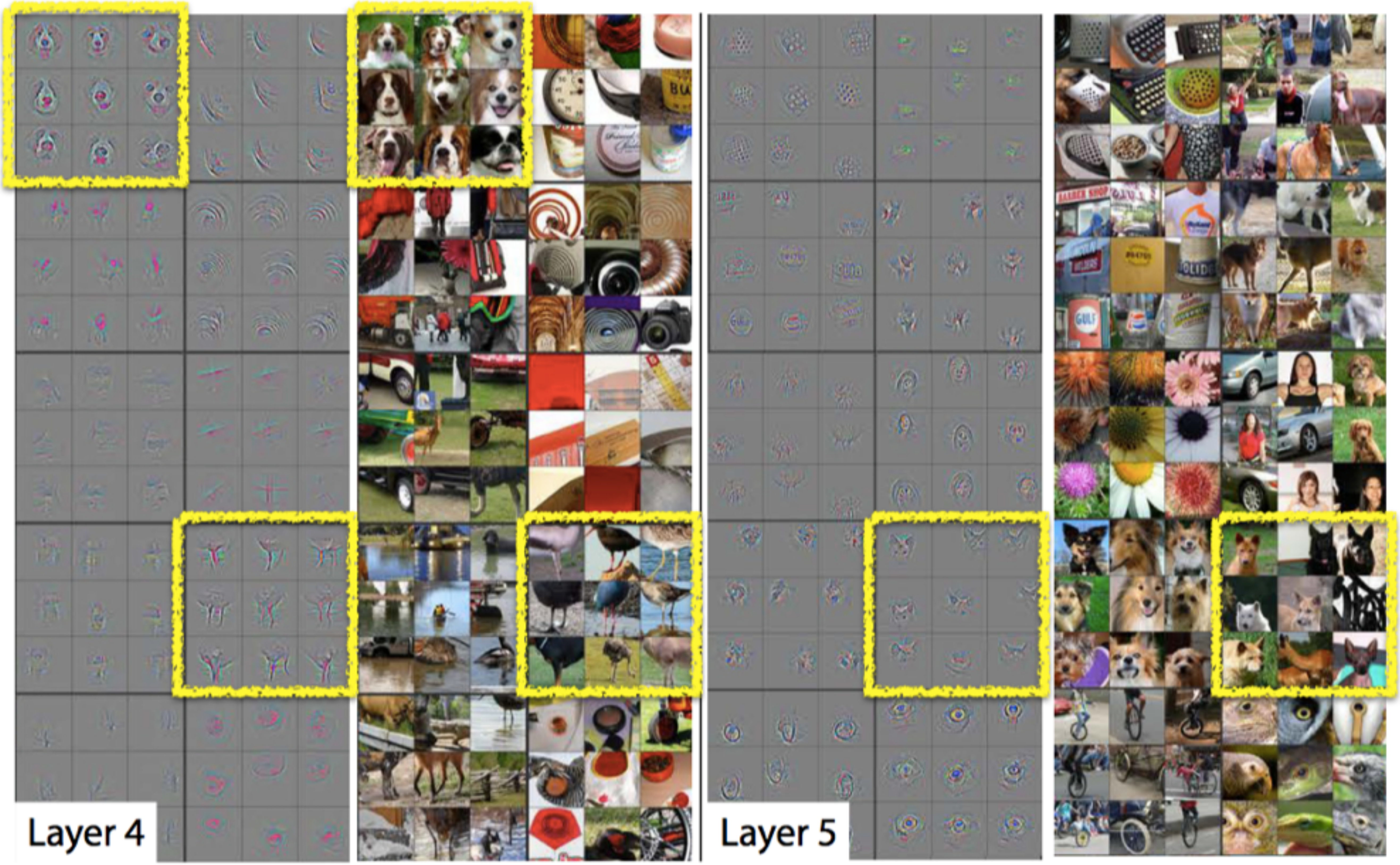


corners & edge/color conjunctions

reverse **projections** of neuron outputs in pixel space



similar textures



Object parts
(dog face & bird legs)

Entire object with pose variation
(dogs)

compositional features

compositional problem solving

multiplication (circuit design)

<— composed of adding numbers

<— composed of adding bits

```
     output: x.y
─────multiply─────
───adding nums────
───adding bits────
   input: x and y
```

human knowledge organisation

find roots of a linear expression

<— composed of setting expression to zero and solving linear equations

<— composed of rearranging terms

```
          output: x = -2
─────(find roots of x+2)───────
─────(set x+2=0)─(solve)───────
───────────────(rearrange)─────
       input: x, +, 2, =, 0
```

deep layers make representation of

knowledge and processes happen with **fewer neurons**!

backpropagation?

What is the **target** against which to minimise error?

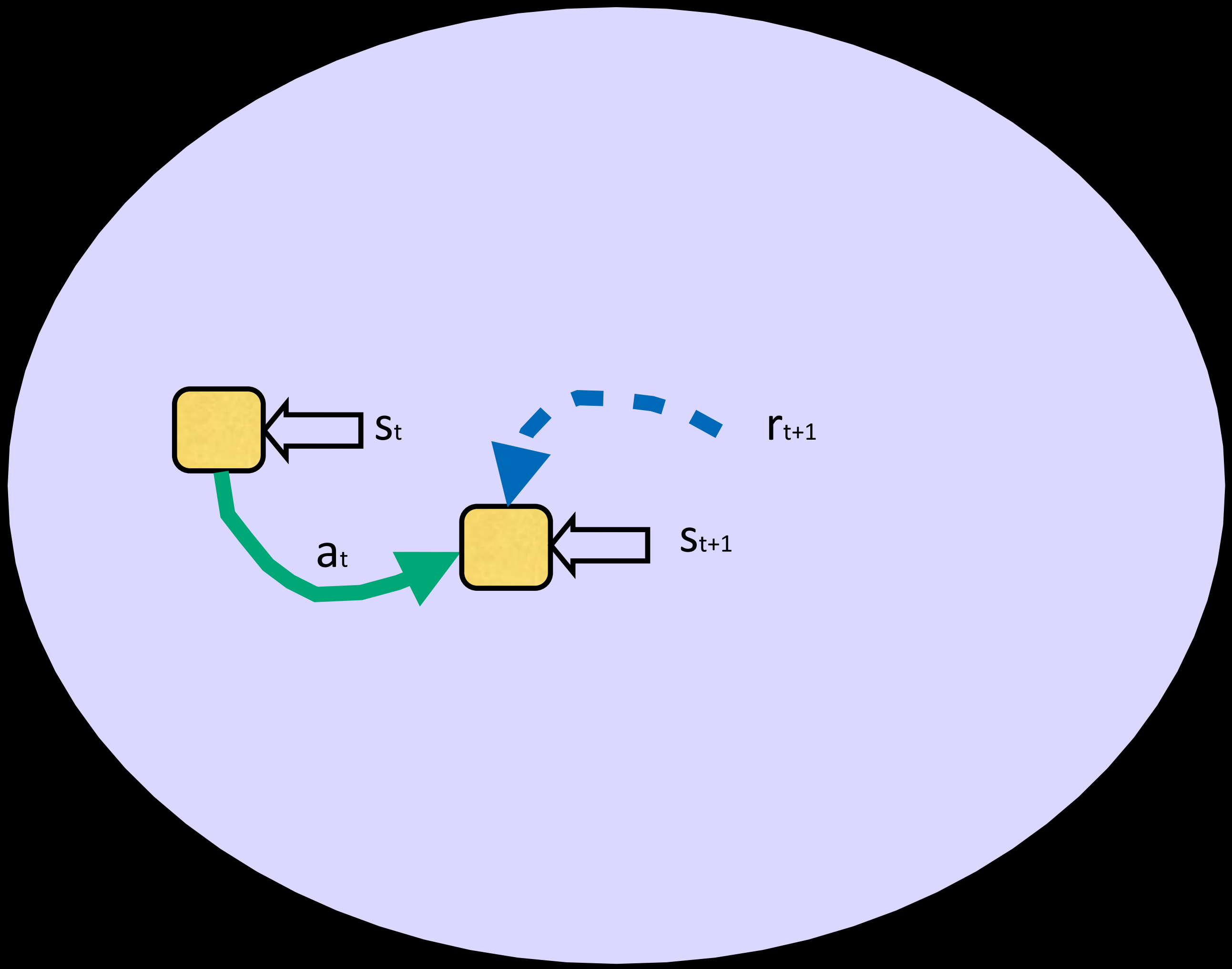
$$\mathcal{L}(w) = \mathbb{E}\left[\left(\underbrace{r + \gamma \max_{a'} Q(s', a', w)}_{\text{target}} - Q(s, a, w)\right)^2\right]$$

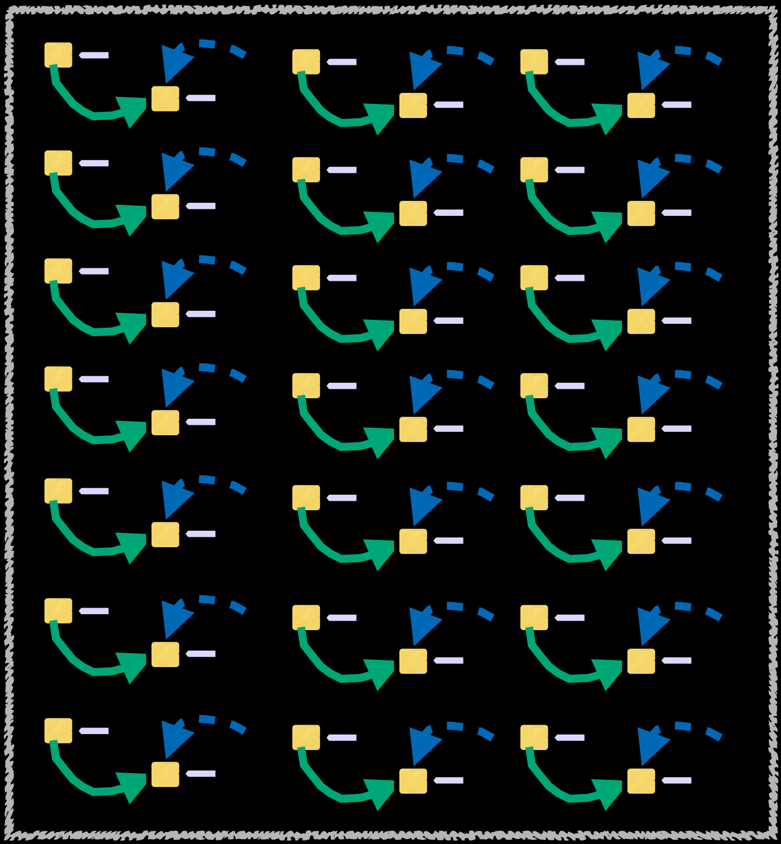
$$\frac{\partial \mathcal{L}(w)}{\partial w} = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s', a', w) - Q(s, a, w)\right) \frac{\partial Q(s, a, w)}{\partial w}\right]$$

practically speaking… minimise MSE by SGD

$$\left(r + \gamma \max_{a'} Q(s', a', \mathbf{w}) - Q(s, a, \mathbf{w}) \right)^2$$

experience replay





save current transition
(s, a, r, s') in memory
every time step

randomly sample a set of
(s, a, r, s') from memory
for training Q network
(instead of learning from current state transition) every step

= i.i.d + learn from the past

freezing target Q

moving target => **oscillations**

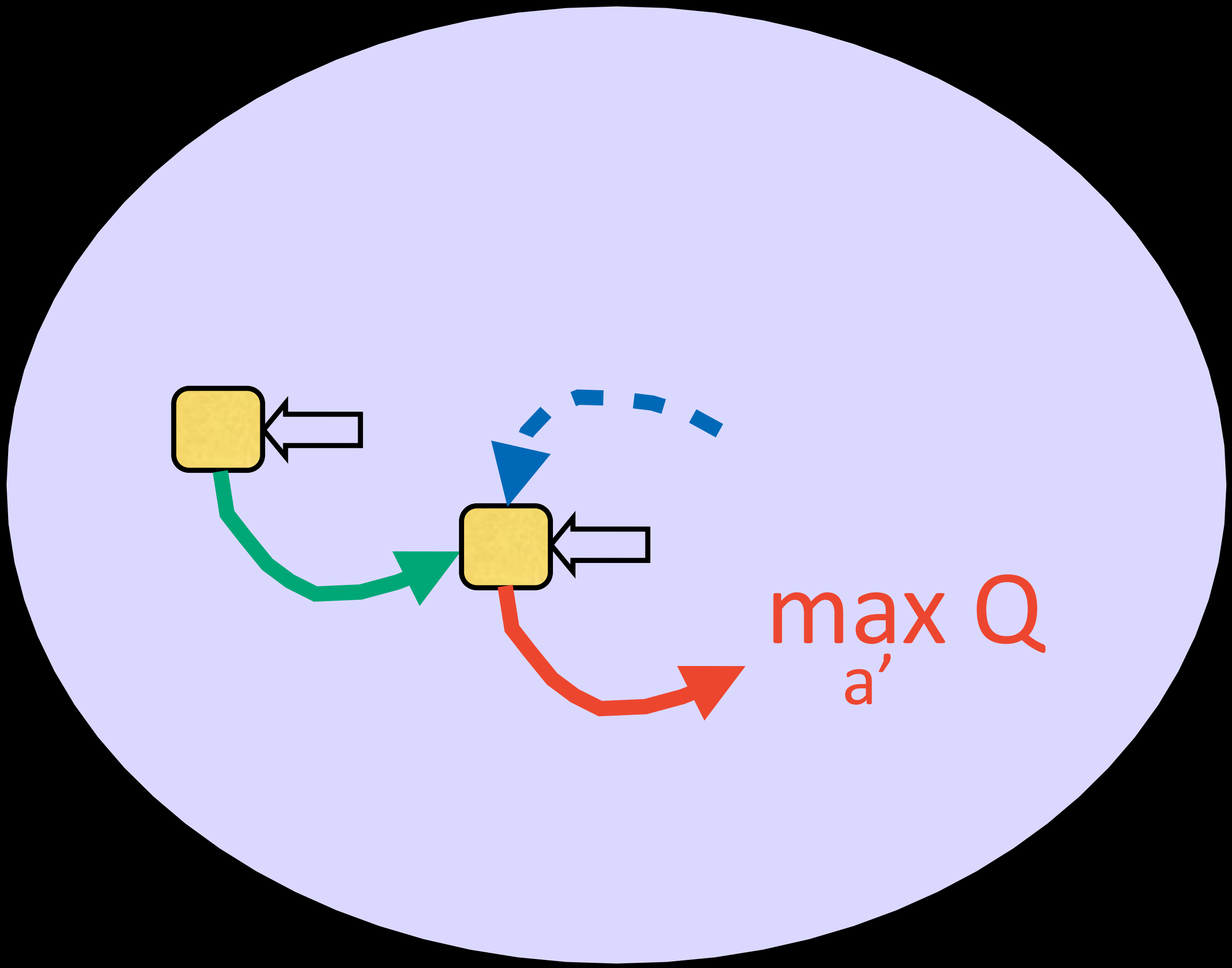
freeze

$$\left(r + \gamma \max_{a'} Q(s', a', \mathbf{w}^-) - Q(s, a, \mathbf{w}) \right)^2$$

stabilise learning by **fixing target**,

moving it every now and then

double DQN



selection of target action

evaluation of target action

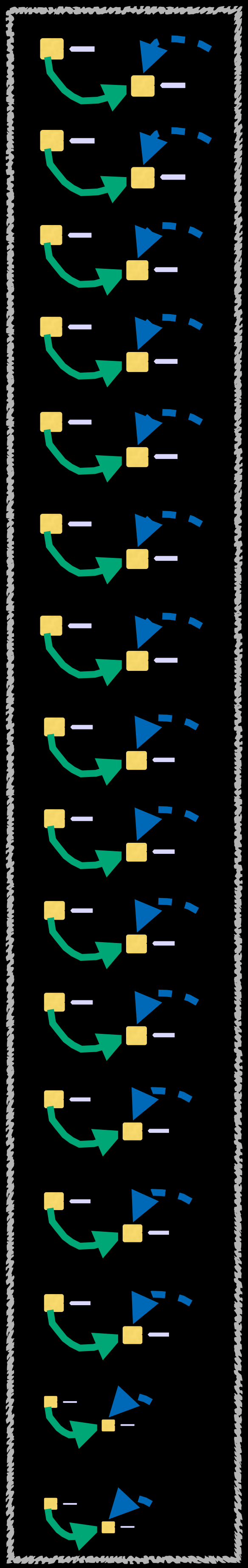
$$\left(r + \gamma Q(s', \underset{a'}{\operatorname{argmax}}\, Q(s', a', \mathbf{w}), \mathbf{w}^-) - Q(s, a, \mathbf{w})\right)^2$$

Deep Reinforcement Learning with Double Q-learning
van Hasselt et. al., AAAI 2016
https://arxiv.org/pdf/1509.06461v3.pdf

prioritised experience replay

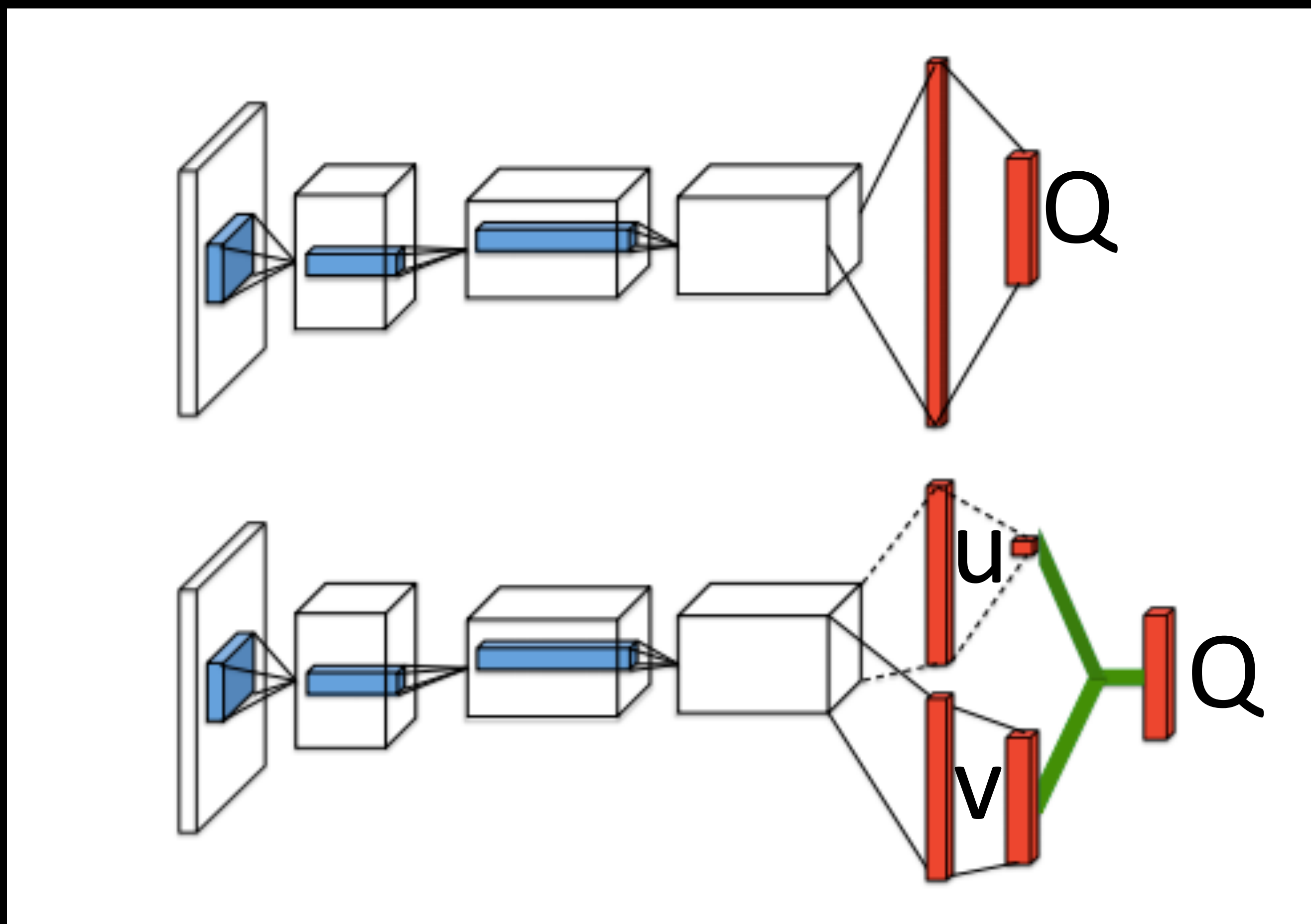
sample
(s, a, r, s’) from memory
based on surprise

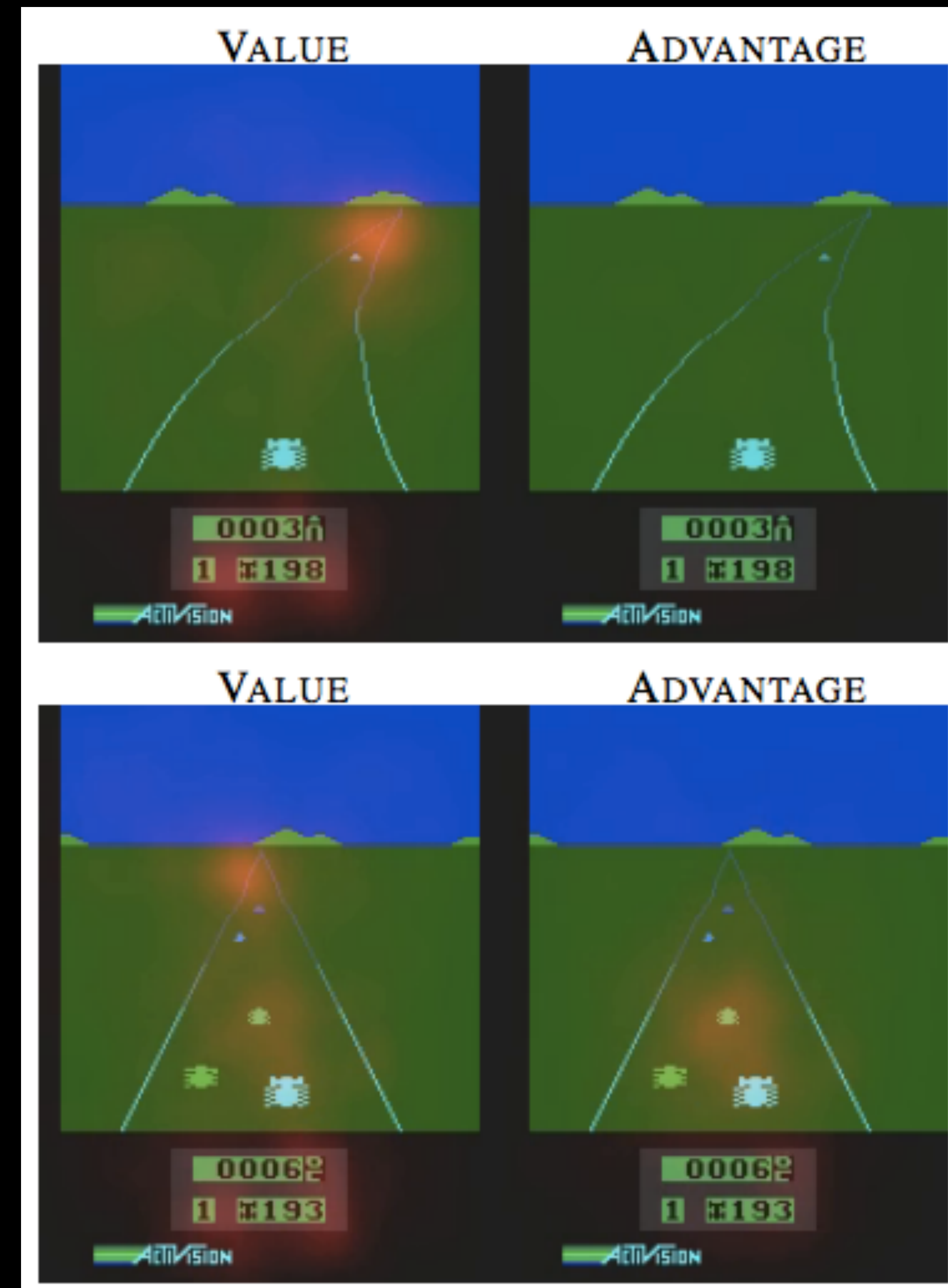
$$\left| r + \gamma \max_{a'} Q(s', a', \mathbf{w}^{-}) - Q(s, a, w) \right|$$



Prioritised Experience Replay
Schaul et. al., ICLR 2016
https://arxiv.org/pdf/1511.05952v4.pdf

duelling architecture





$$Q(s, a) = V(s, u) + A\,(s, a, v)$$

Dueling Network Architectures for Deep RL
Wang et. al., ICML 2016
https://arxiv.org/pdf/1511.06581v3.pdf

Combining decoupling (double), prioritised replay, and duelling helps!

however training is

SLOW

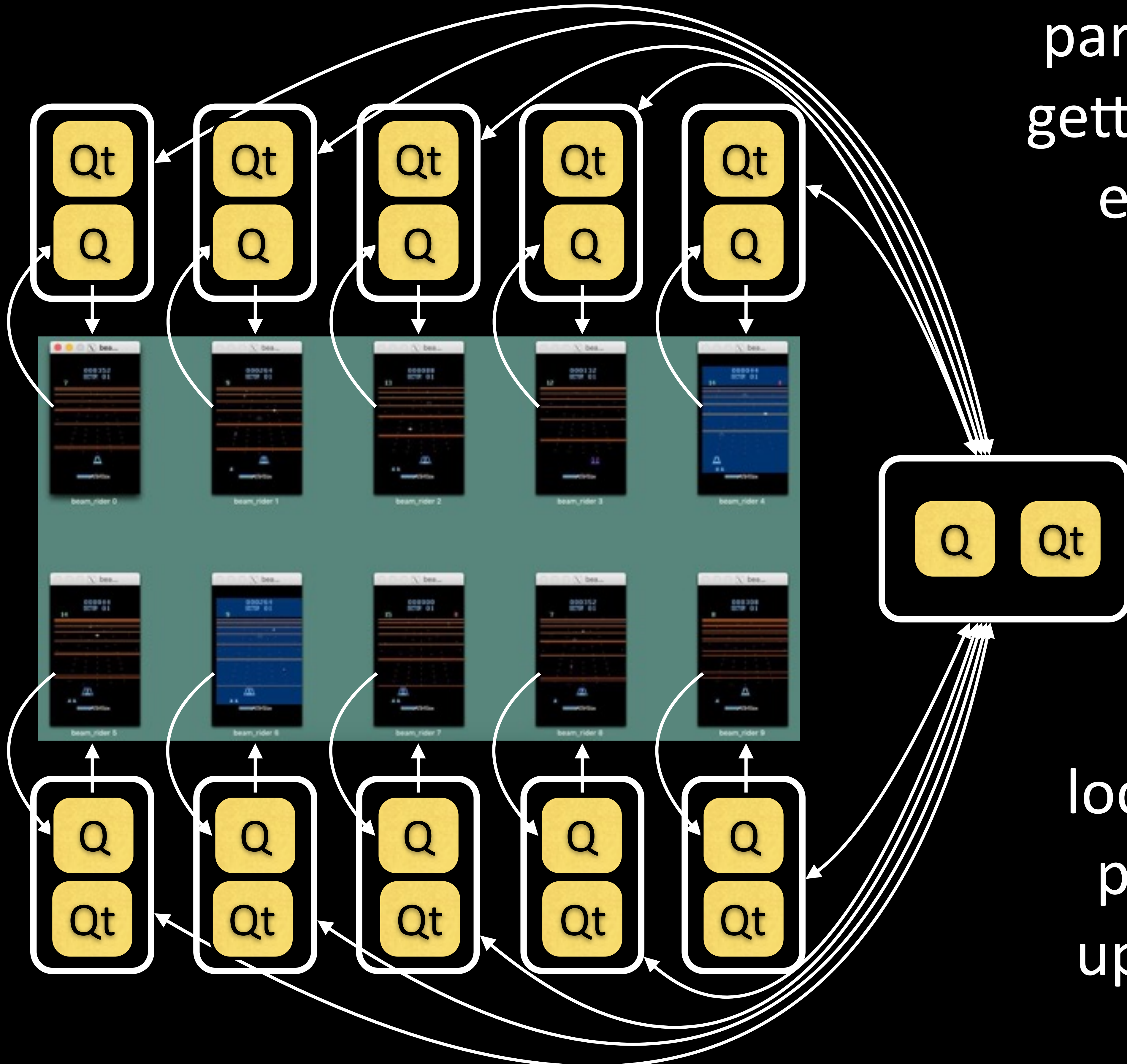
making deep RL faster and wilder (more applicable in the real world)!

data efficient exploration?

making use of a model?

transfer learning?

parallelism?



parallel learners getting individual experiences

shared params for Q and target Q

lock-free param updates

Asynchronous Methods for Deep Reinforcement Learning, Mnih et. al., ICML 2016
http://jmlr.org/proceedings/papers/v48/mniha16.pdf

code for you to play with…

Telenor's own implementation of **asynchronous deep RL**:
https://github.com/traai/async-deep-rl

OpenAI Gym BETA

Let's keep the conversation going: https://openrl.slack.com