

# scaling up RL with function approximation

[arjun.chandra@telenor.com](mailto:arjun.chandra@telenor.com)

# human level game control

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

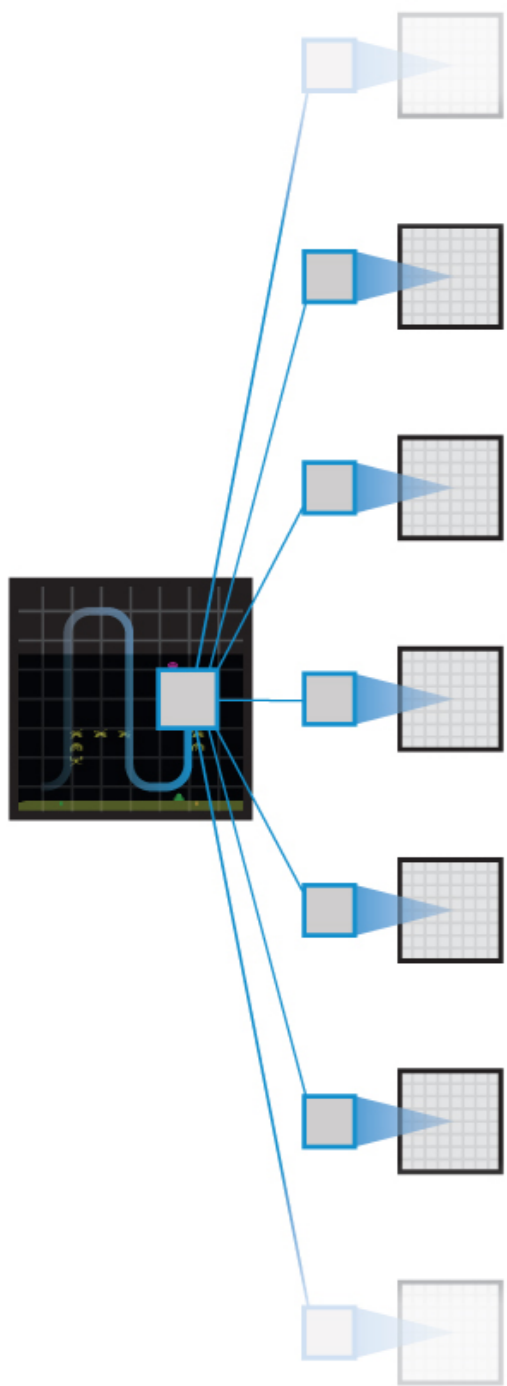
Human-level control through deep reinforcement learning,

Mnih et. al., Nature 518, Feb 2015

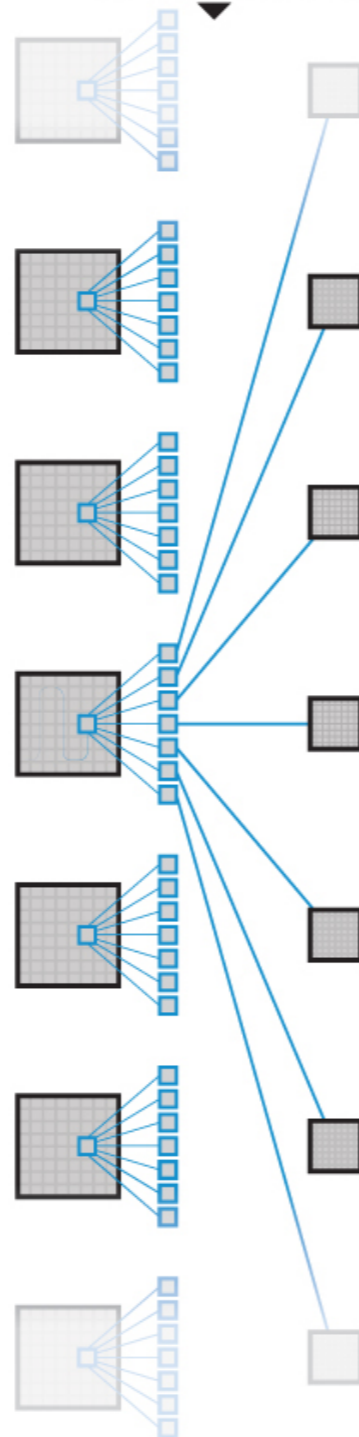
<http://www.nature.com/nature/journal/v518/n7540/full/nature14236.html>

# neural network

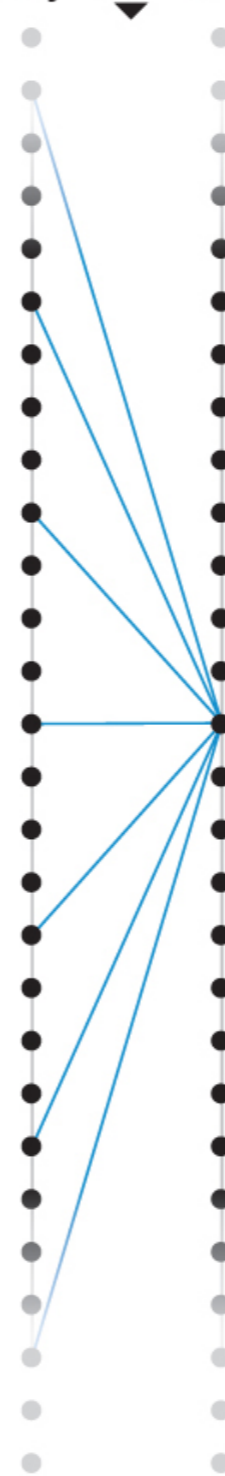
Convolution



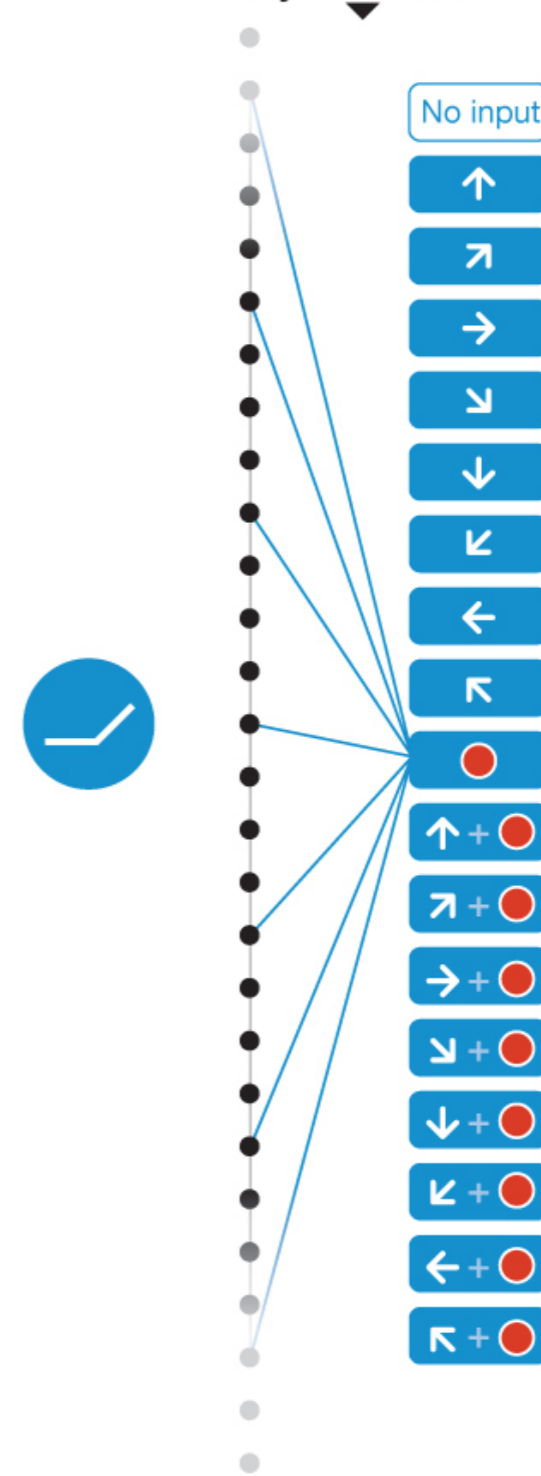
Convolution



Fully connected

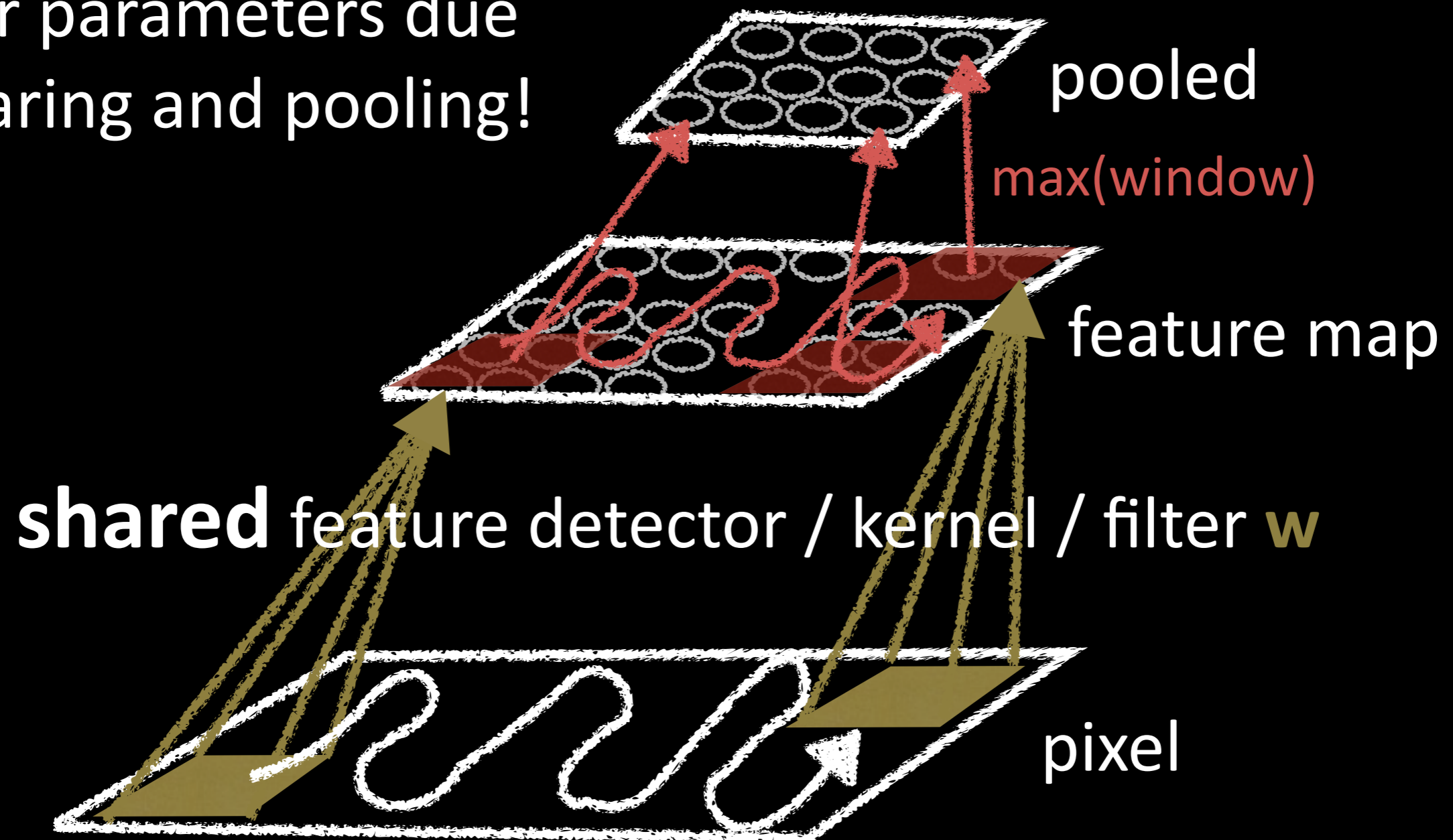


Fully connected

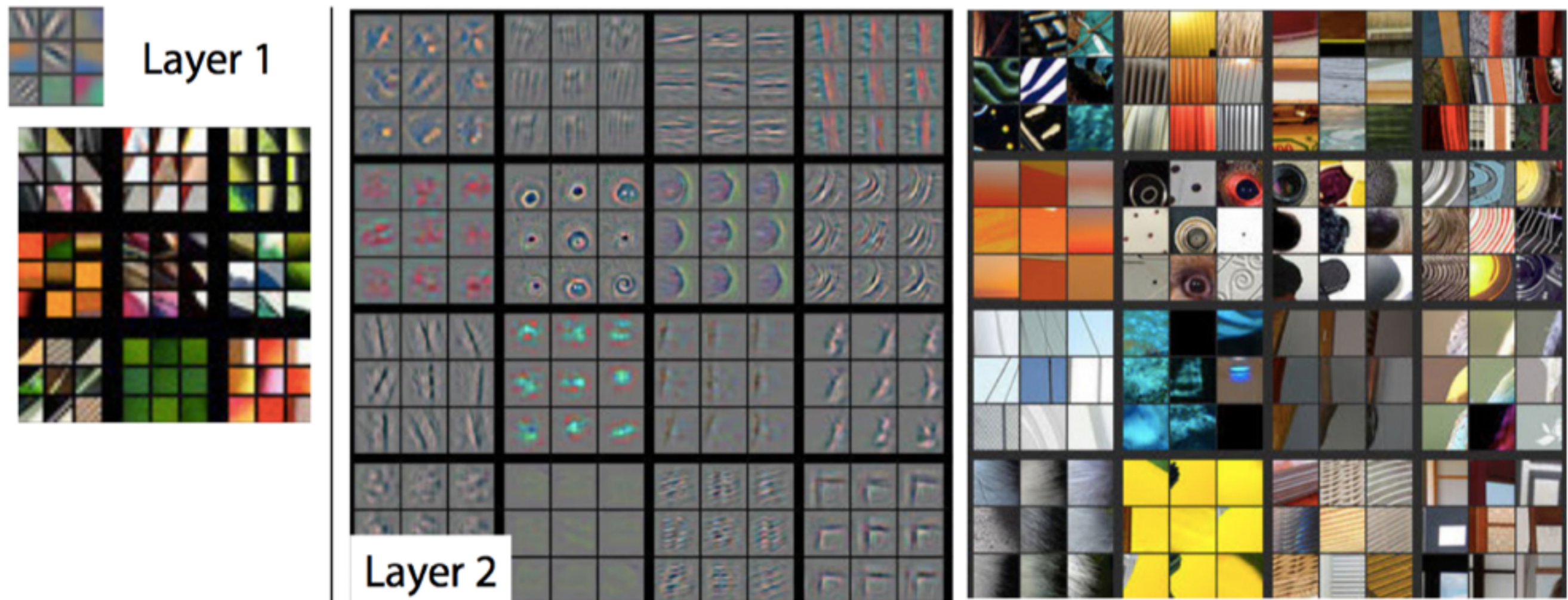


# convolution, weight sharing, and pooling

fewer parameters due to sharing and pooling!

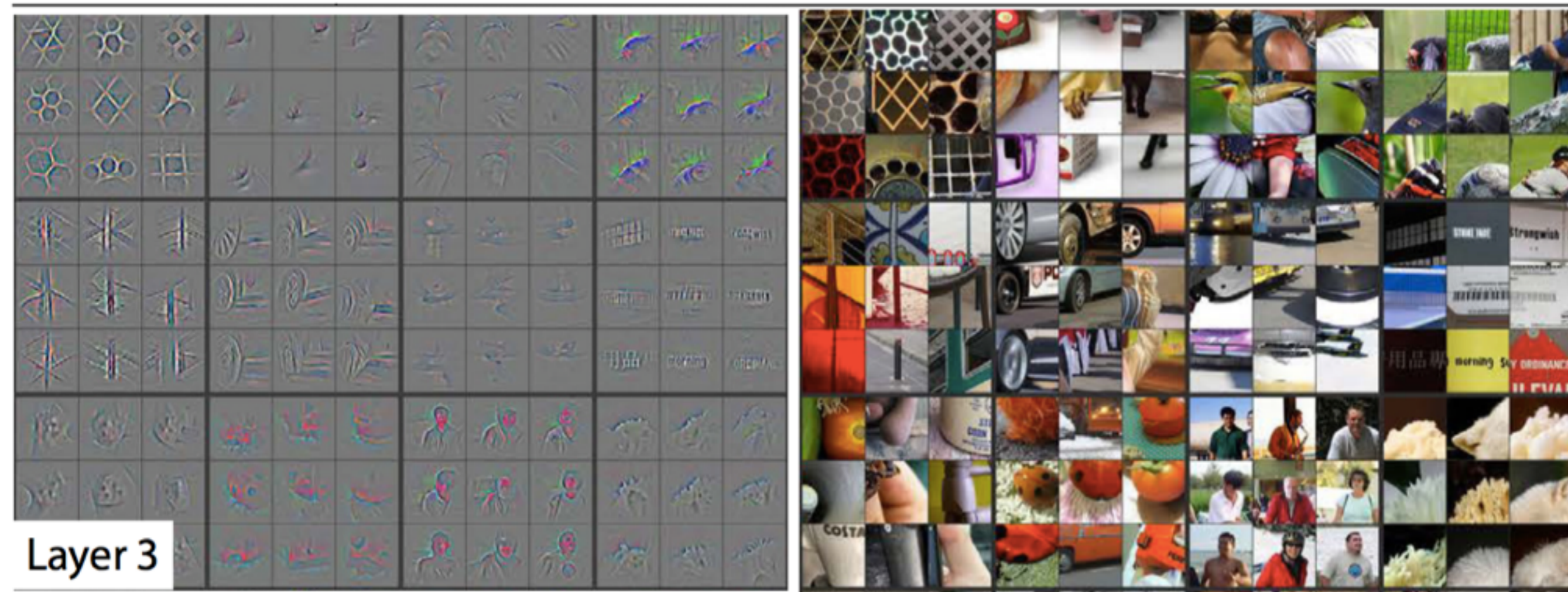


# what does a deep neural network do?



corners & edge/color conjunctions

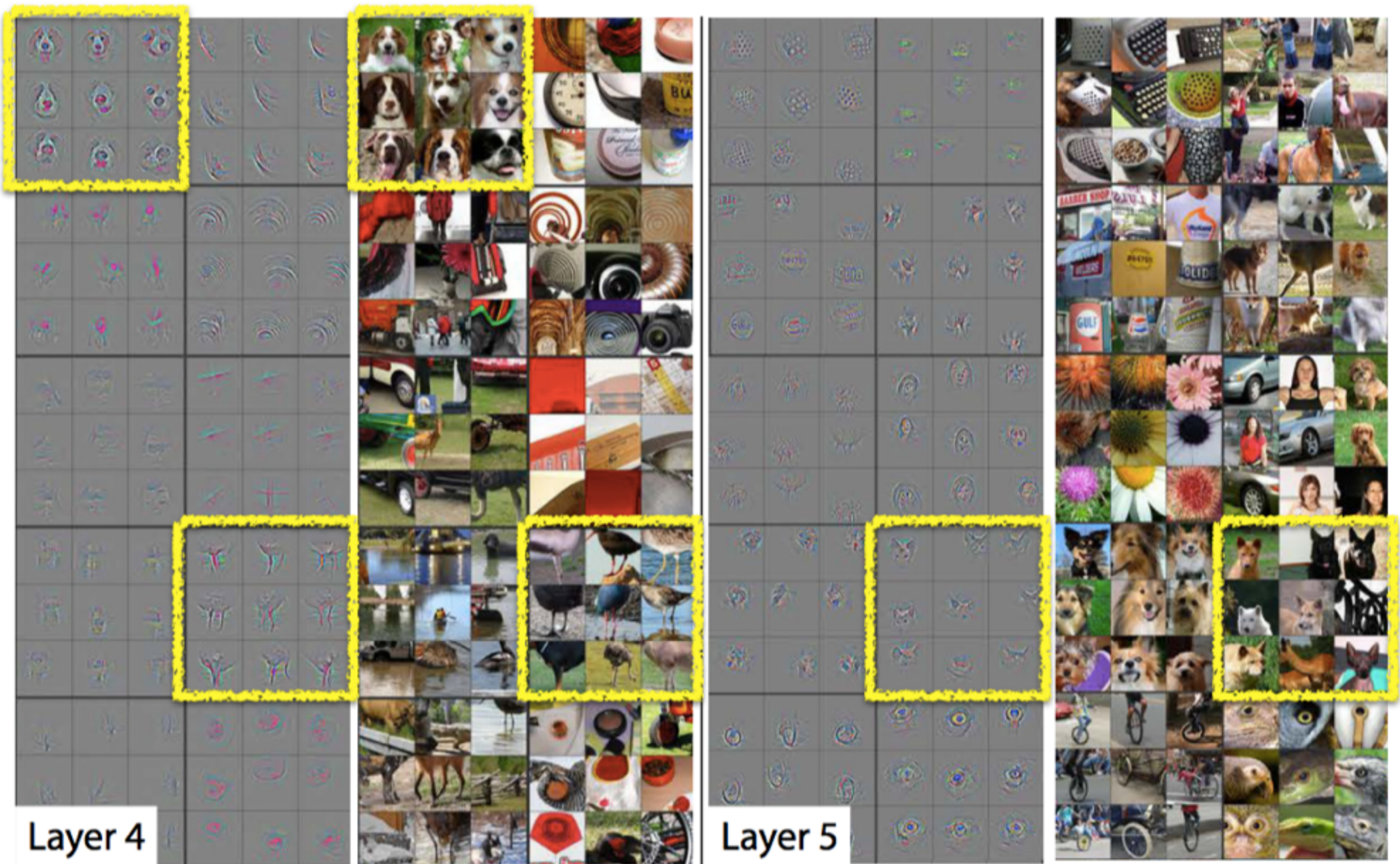
reverse projections of neuron outputs in pixel space



Layer 3



similar textures



Layer 4

Layer 5

Object parts  
(dog face & bird legs)

Entire object with pose variation  
(dogs )

# backpropagation?

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

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

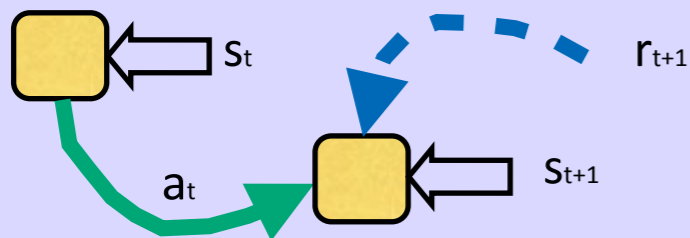
$$\frac{\partial \mathcal{L}(w)}{\partial w} = \mathbb{E} \left[ \left( r + \gamma \max_{a'} Q(s', a', w) - Q(s, a, w) \right) \frac{\partial Q(s, a, w)}{\partial w} \right]$$



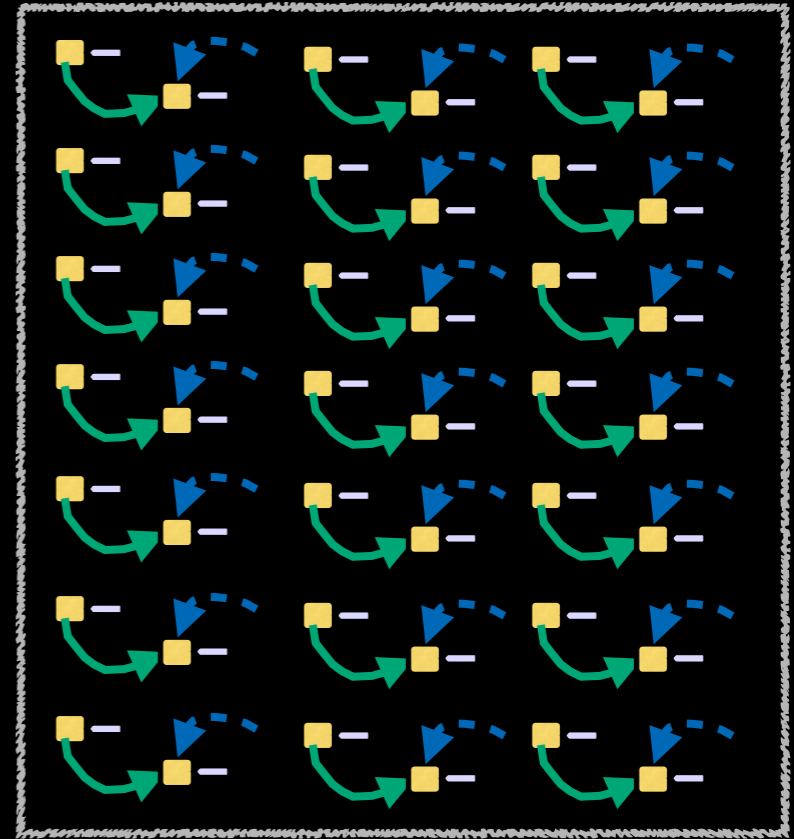
practically speaking...  
minimise MSE by SGD

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

# experience replay



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



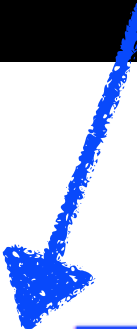
randomly sample a set of  
 $(s, a, r, s')$  from memory  
for training Q network  
**(instead of learning from  
current state transition)**  
every step

= i.i.d + learn from the past

# freezing target Q

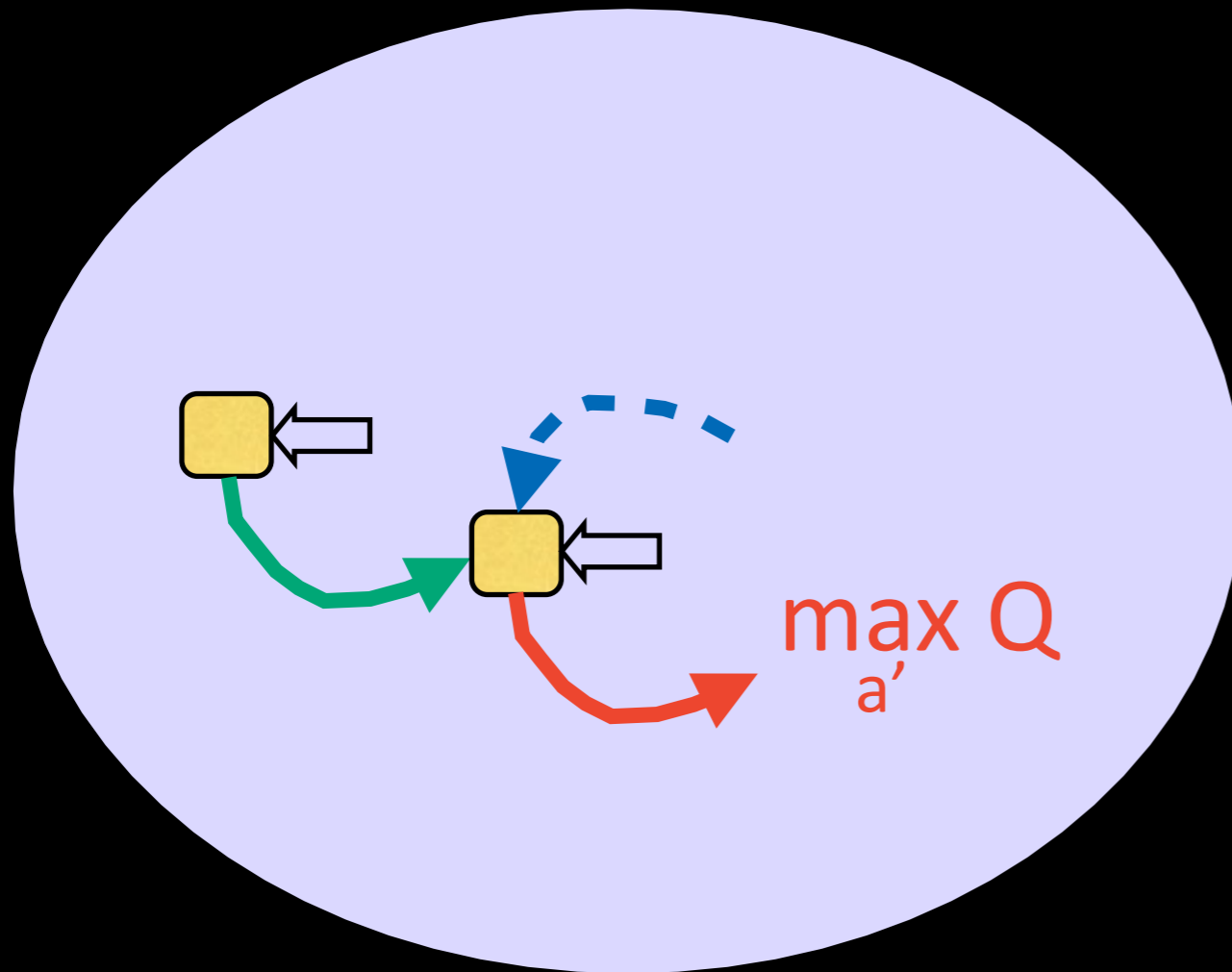
moving target => **oscillations**

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

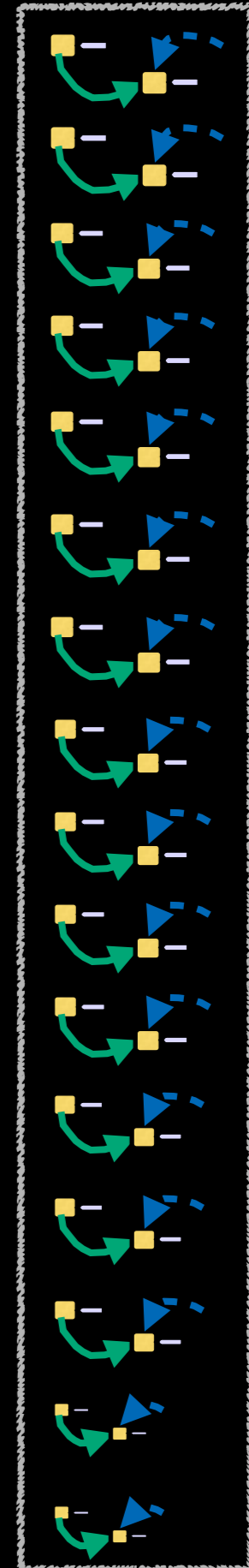
evaluation  
of target action

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

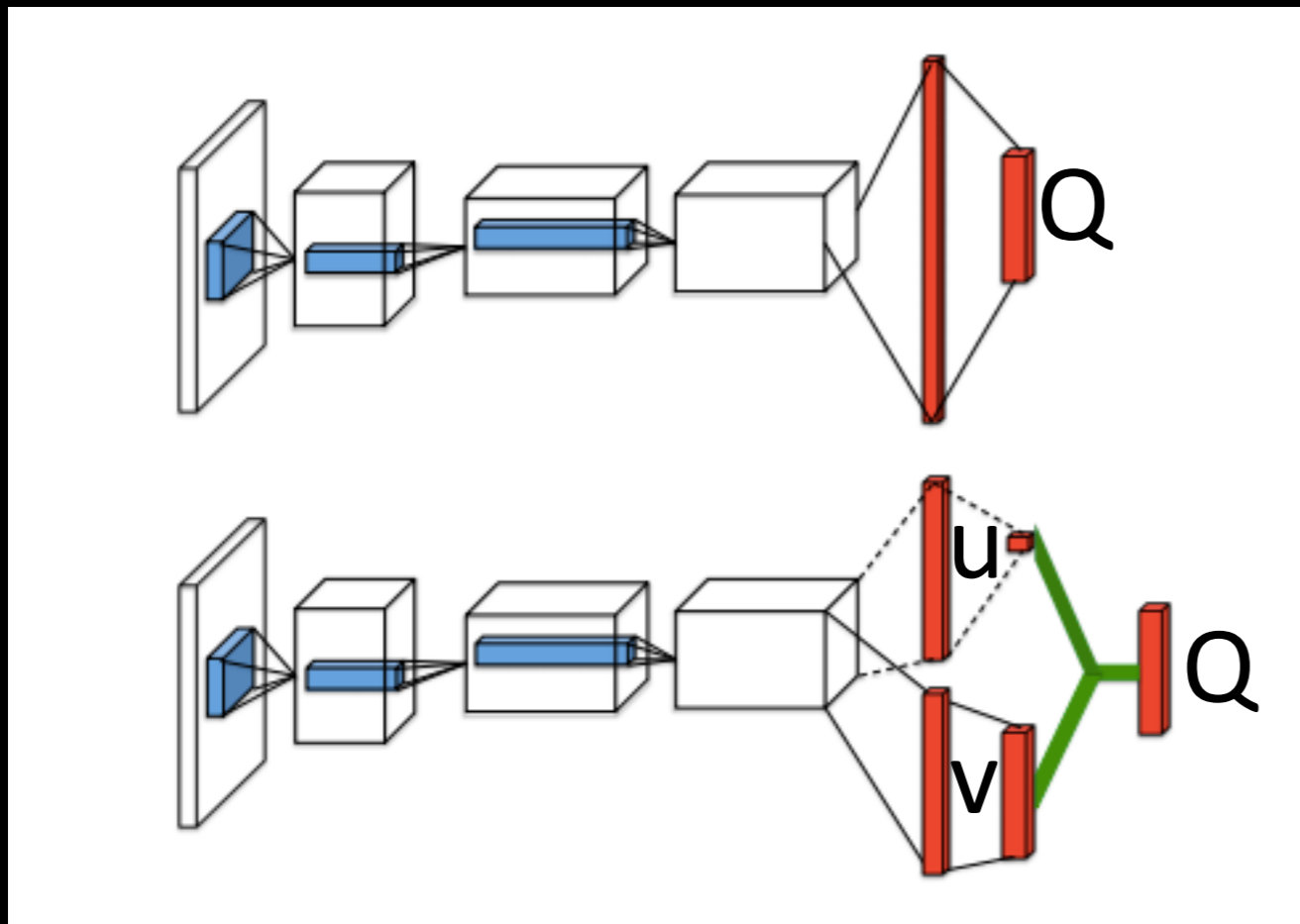
# prioritised experience replay

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

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

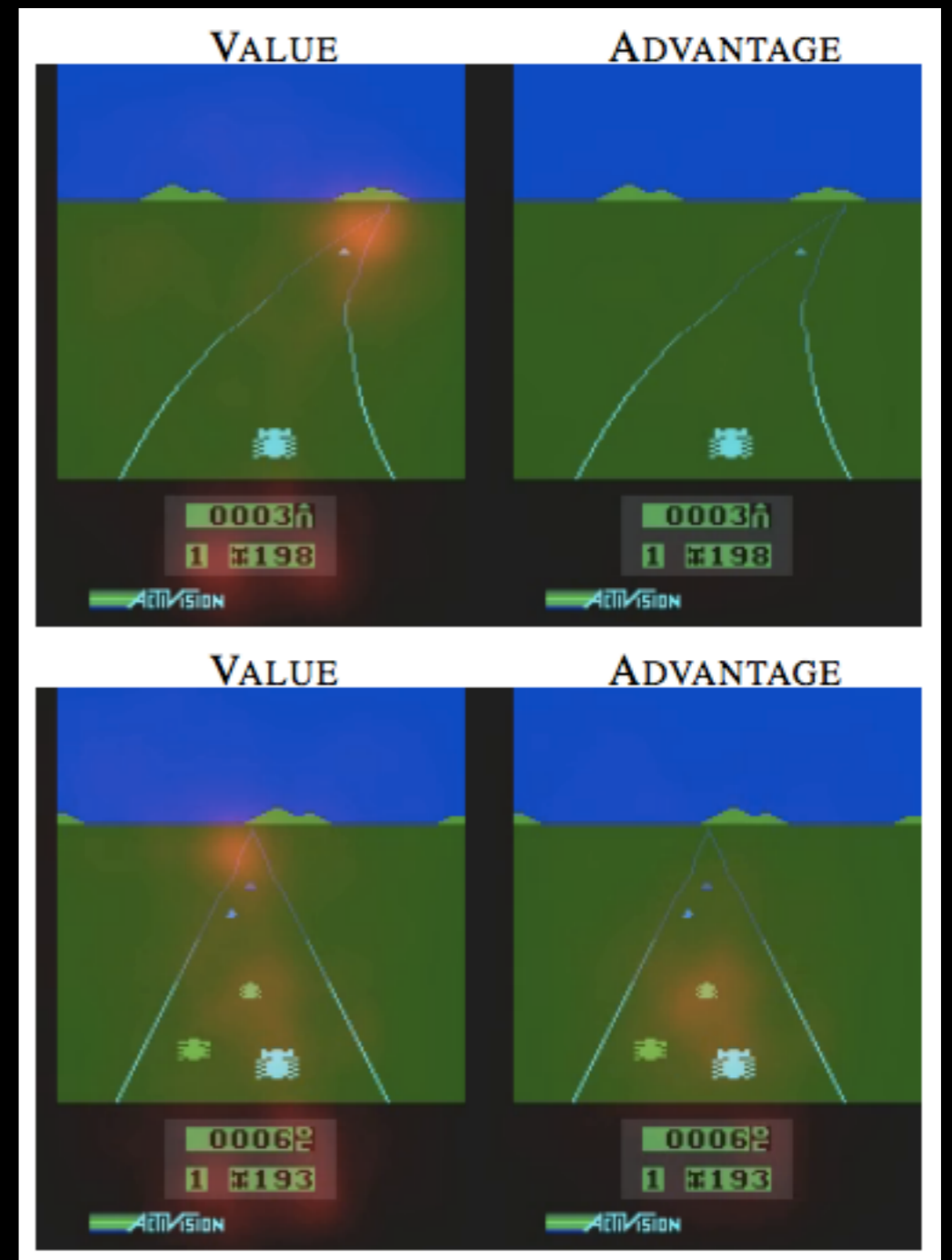


# duelling architecture



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

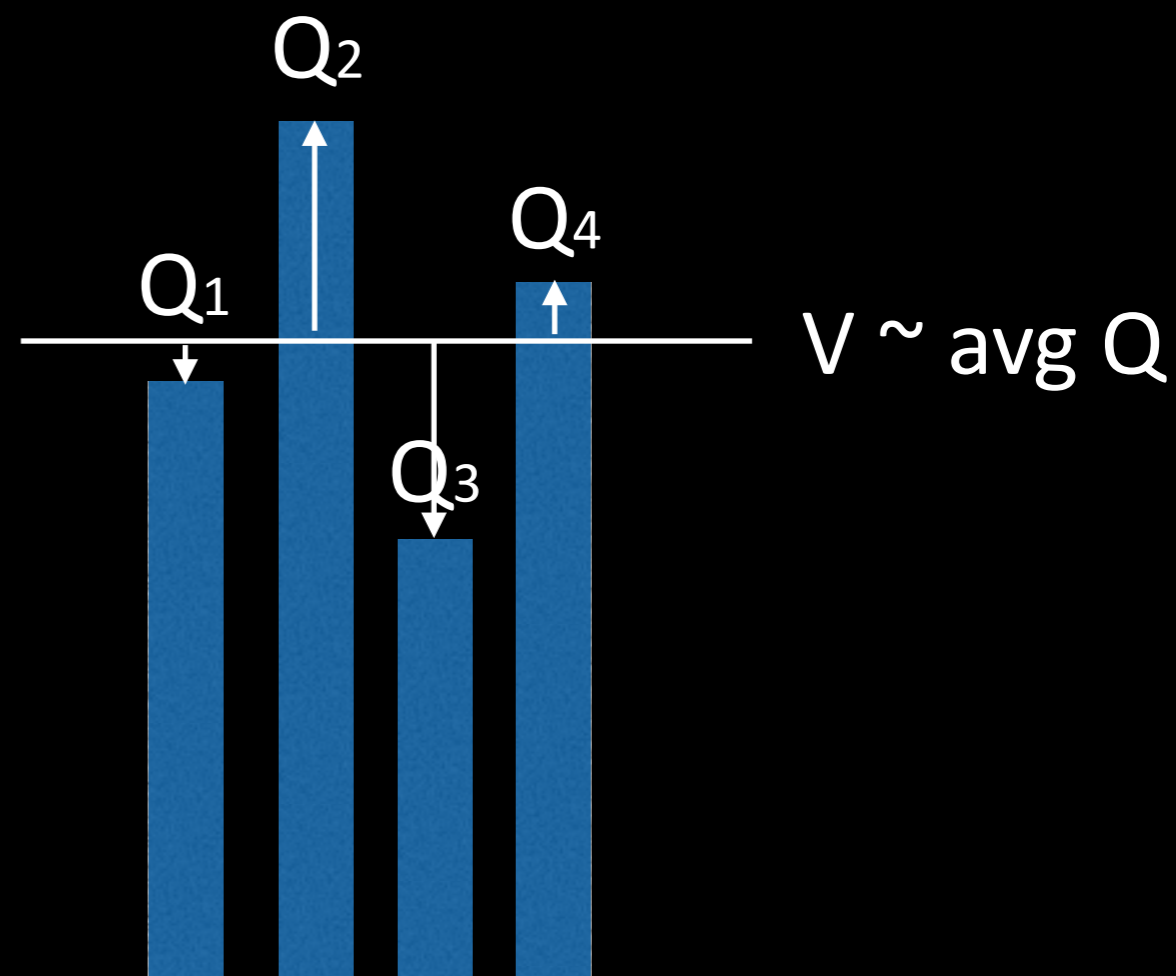
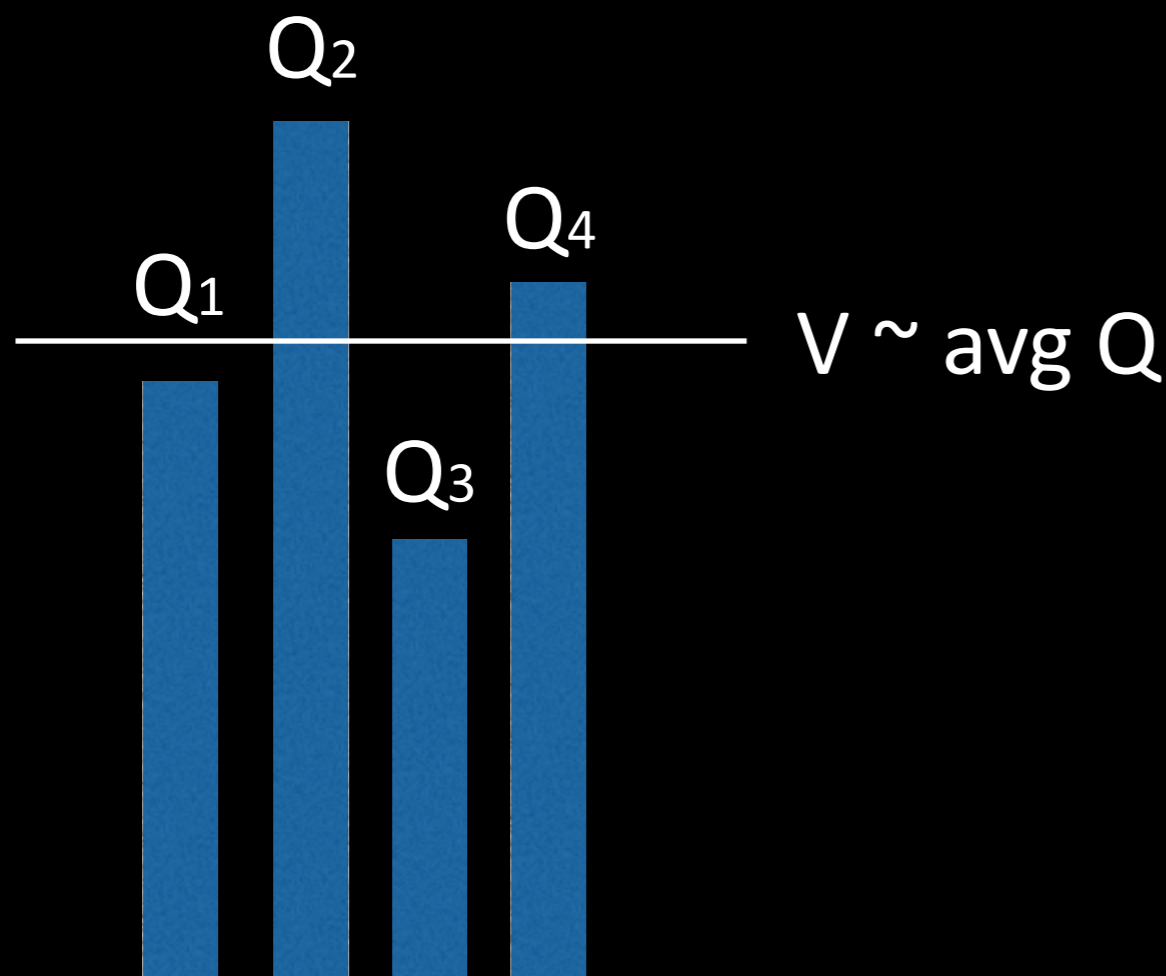
↑  
action a's advantage



advantage?

e.g. 4 actions,

$V = E[Q] \sim Q$  averaged across actions



arrows represent  
advantage of an action

however training is

**SLOW**



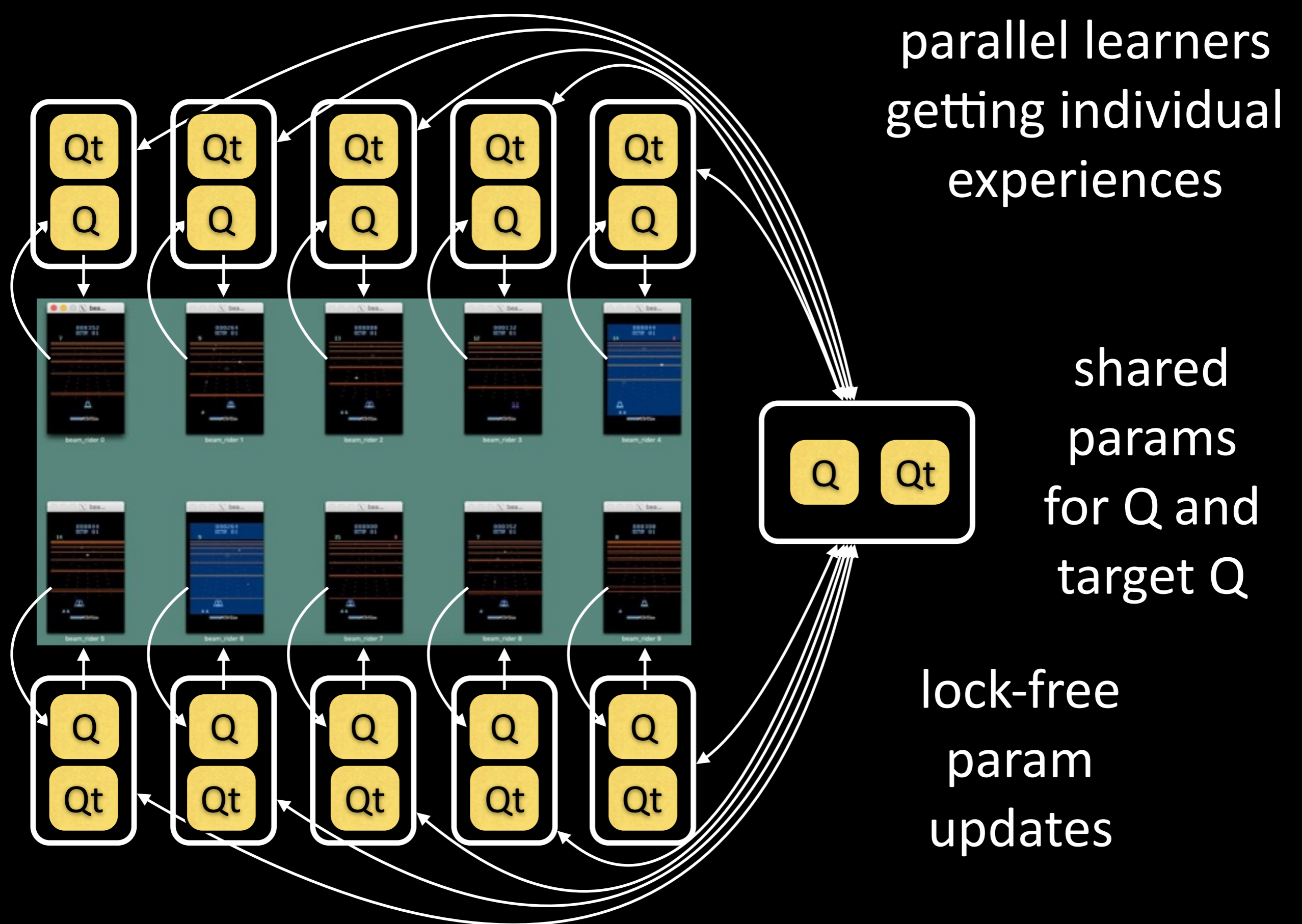
making deep RL faster  
and wilder (more  
applicable in the real  
world)!

data efficient exploration?

making use of a model?

transfer learning?

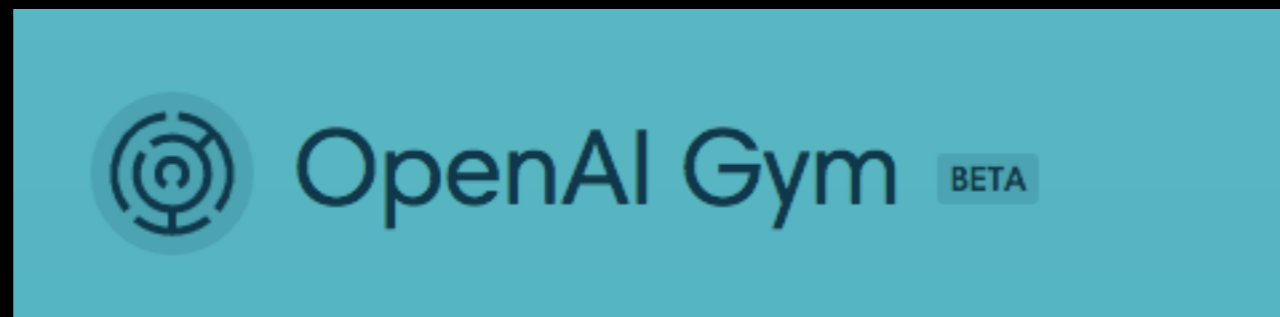
**parallelism?**



# code for you to play with...

Telenor's own implementation of **asynchronous deep RL**:

<https://github.com/traai/async-deep-rl>



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

Join using your [student.matnat.uio.no](mailto:student.matnat.uio.no) email address!

intern with us in summer 2017

CV and motivation to:

[arjun.chandra@telenor.com](mailto:arjun.chandra@telenor.com) | [tina@telenordigital.com](mailto:tina@telenordigital.com)

**(2 positions)**

topic still to be finalised,  
but it will be at the intersection of software  
engineering and machine learning