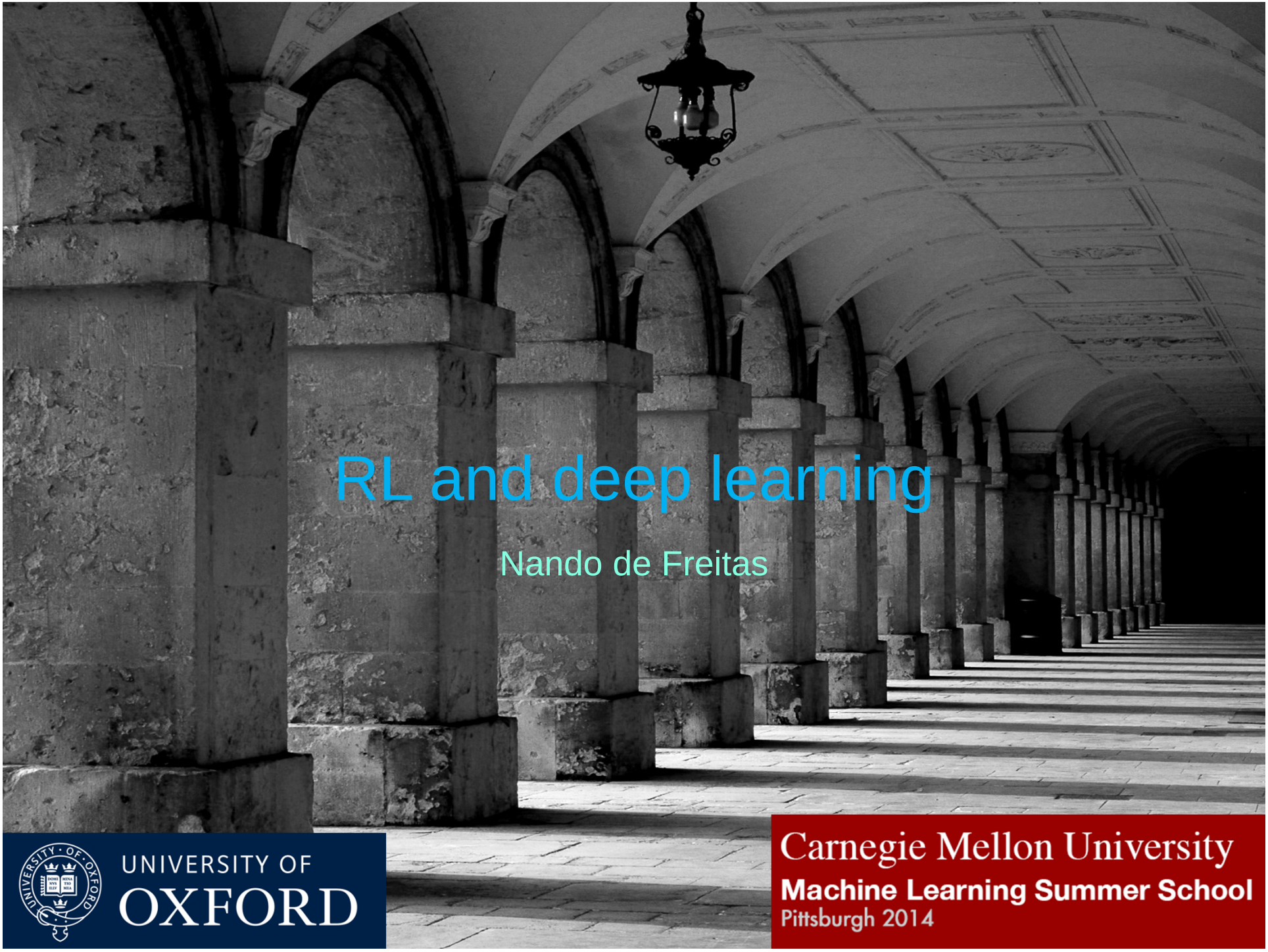
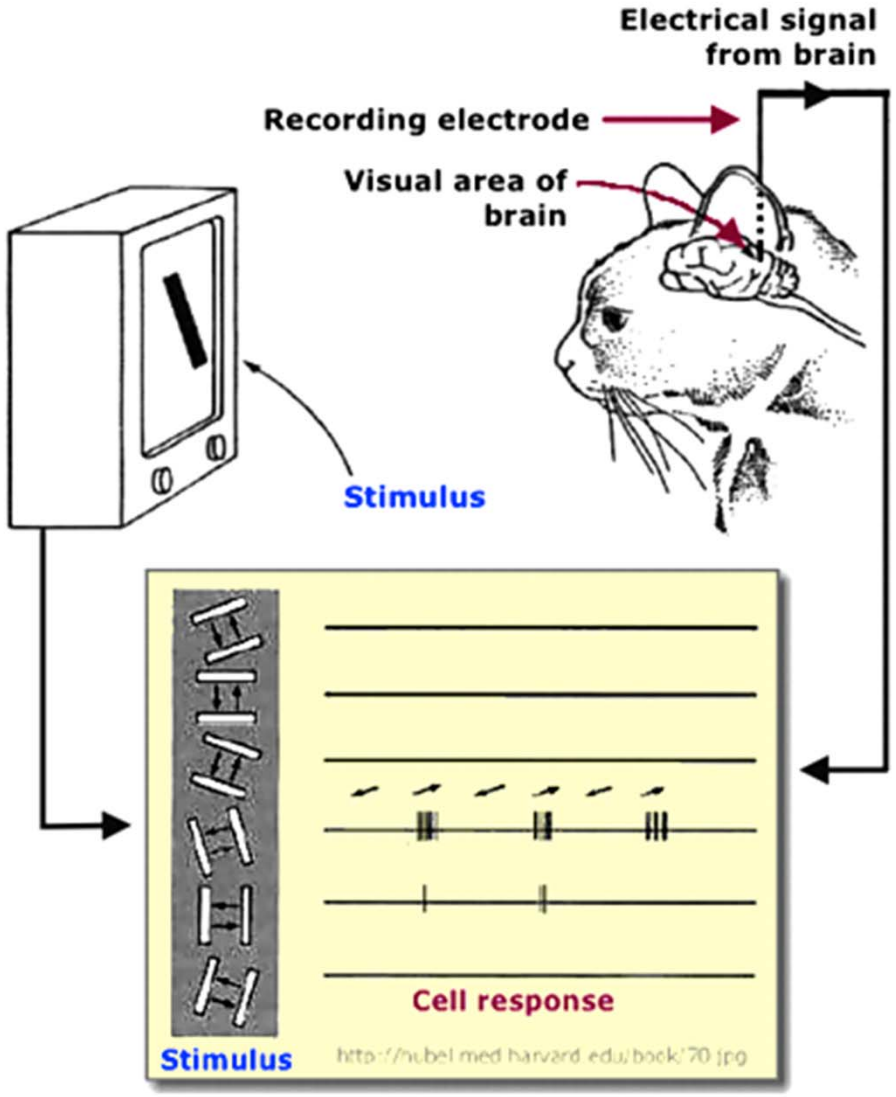


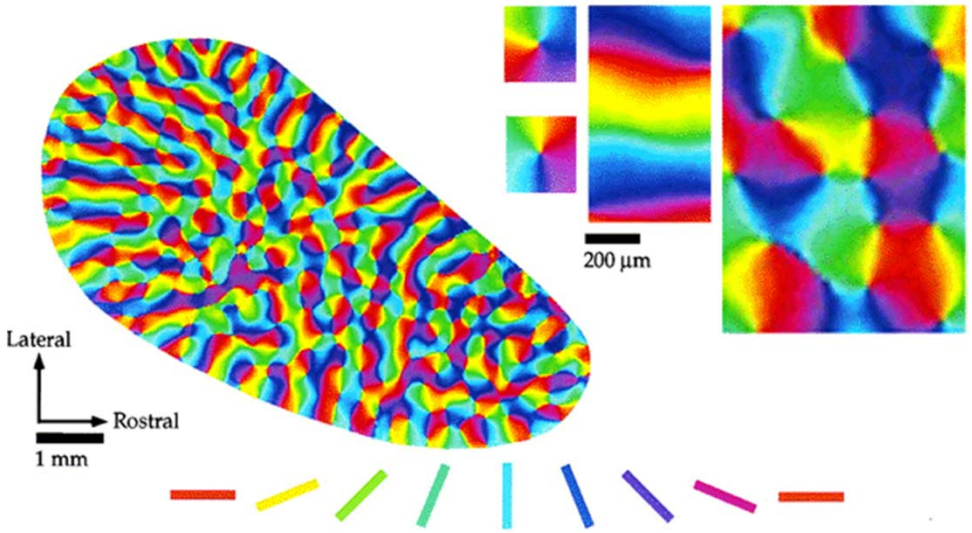
RL and deep learning

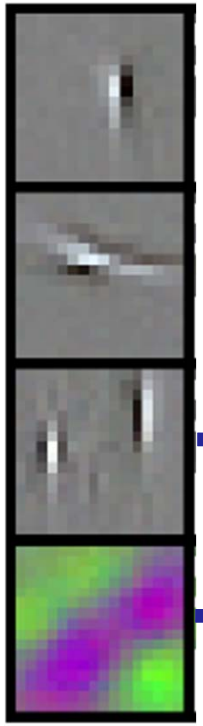
Nando de Freitas

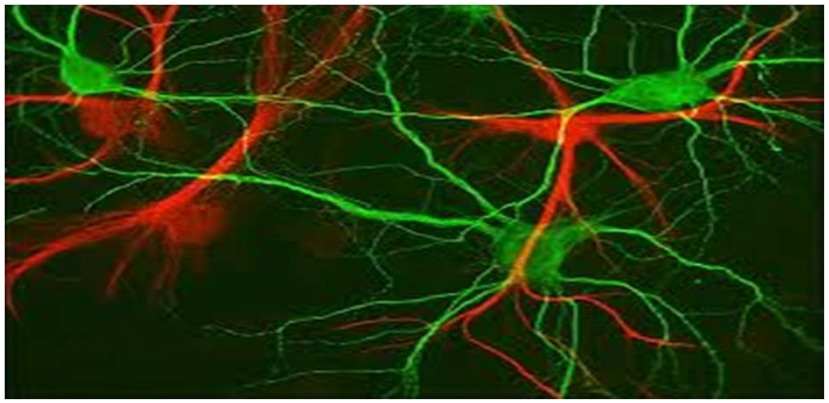
UNIVERSITY OF OXFORD

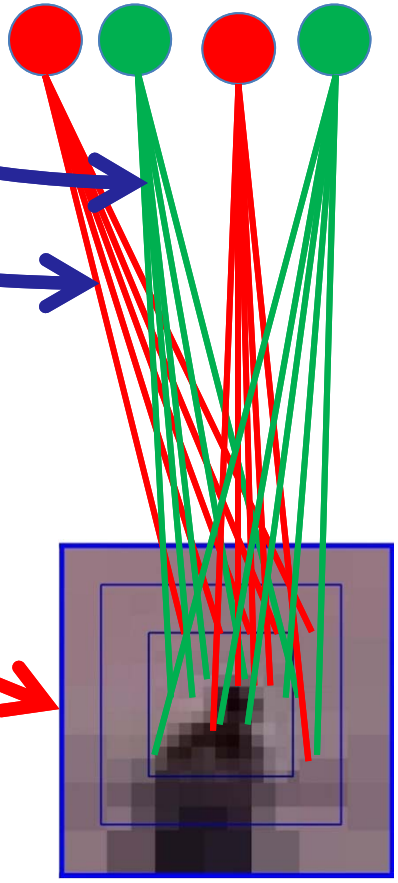
Carnegie Mellon University
Machine Learning Summer School
Pittsburgh 2014


Electrical signal
from brain
Recording electrode
Visual area of
brain
Stimulus
Cell response
Stimulus
http://hubel.med.harvard.edu/book/f70.jpg

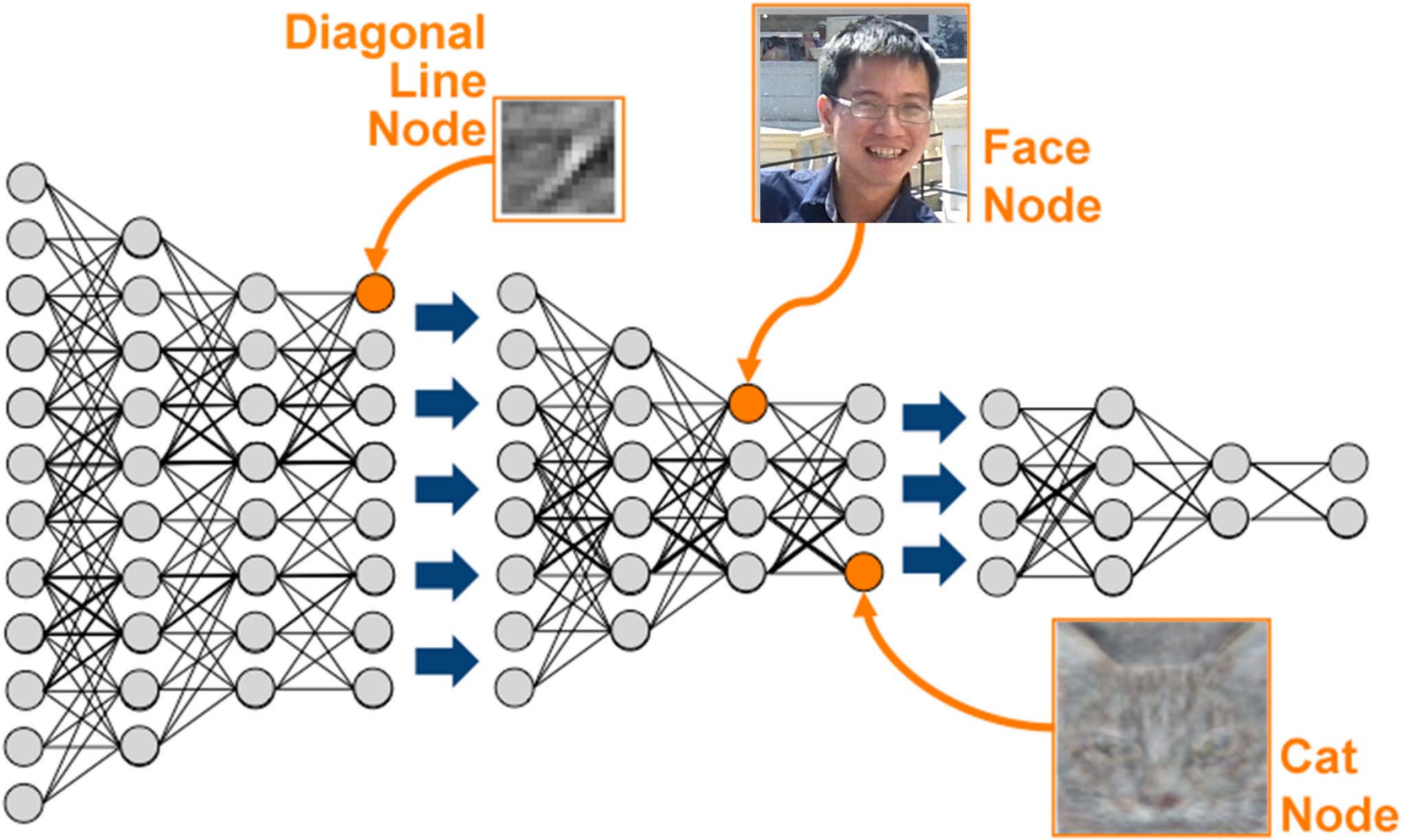

200 µm
Lateral
Rostral
1 mm



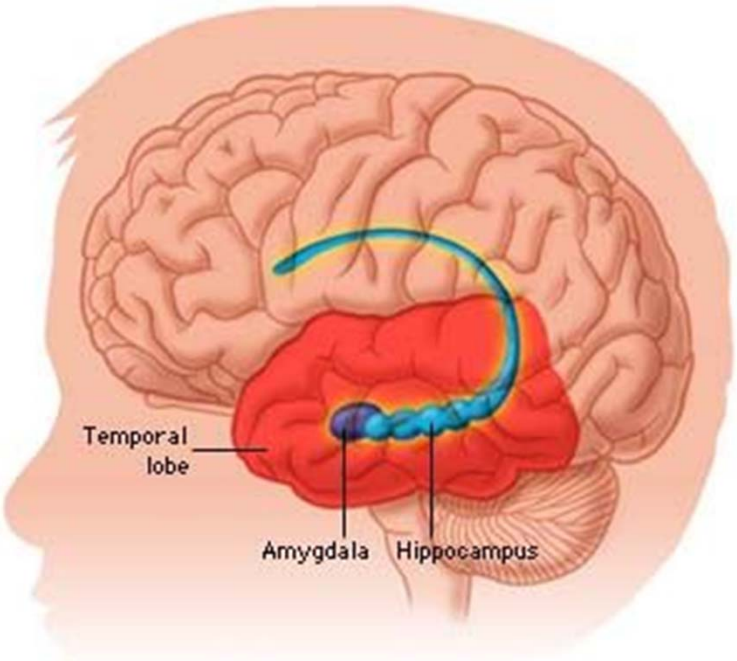


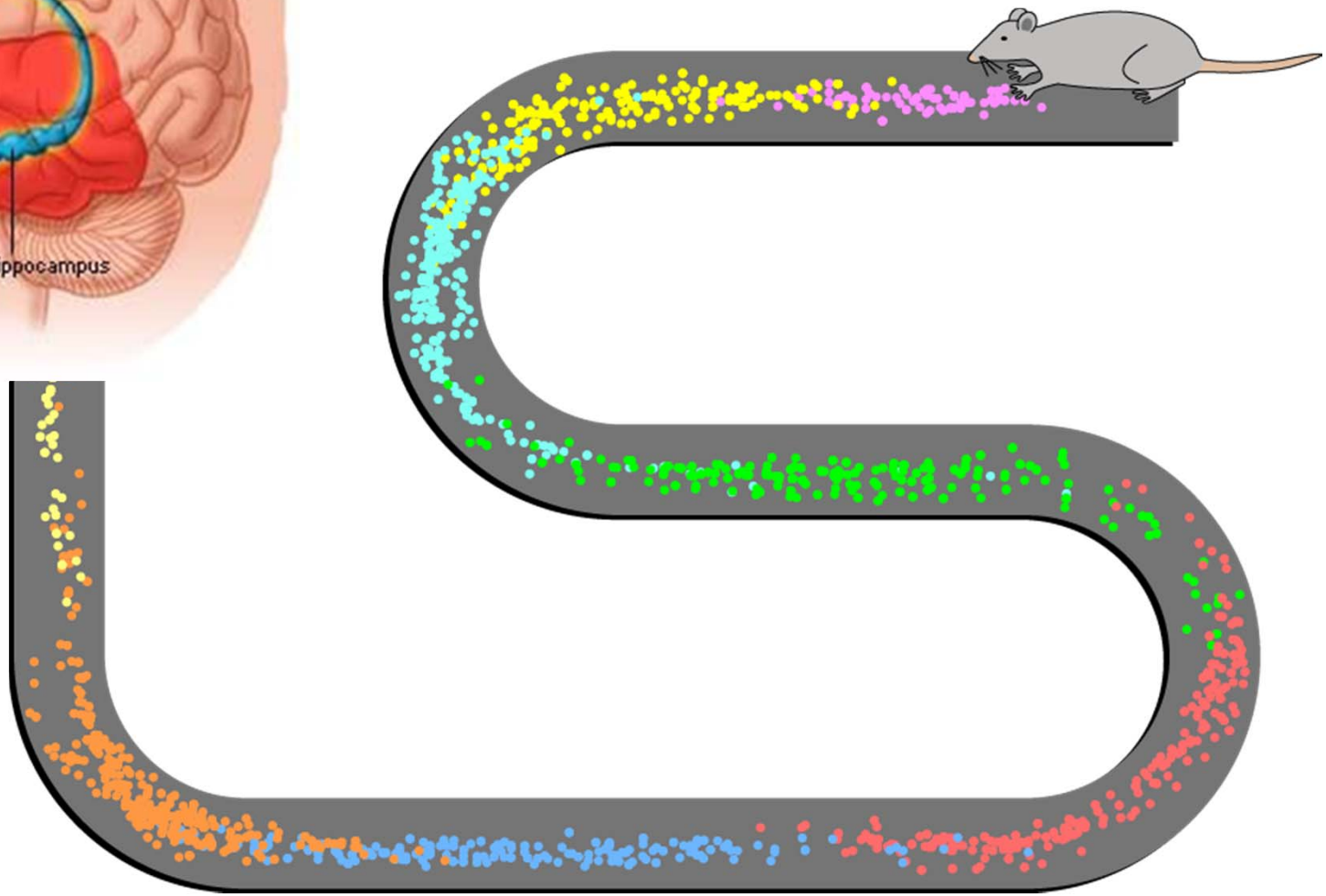


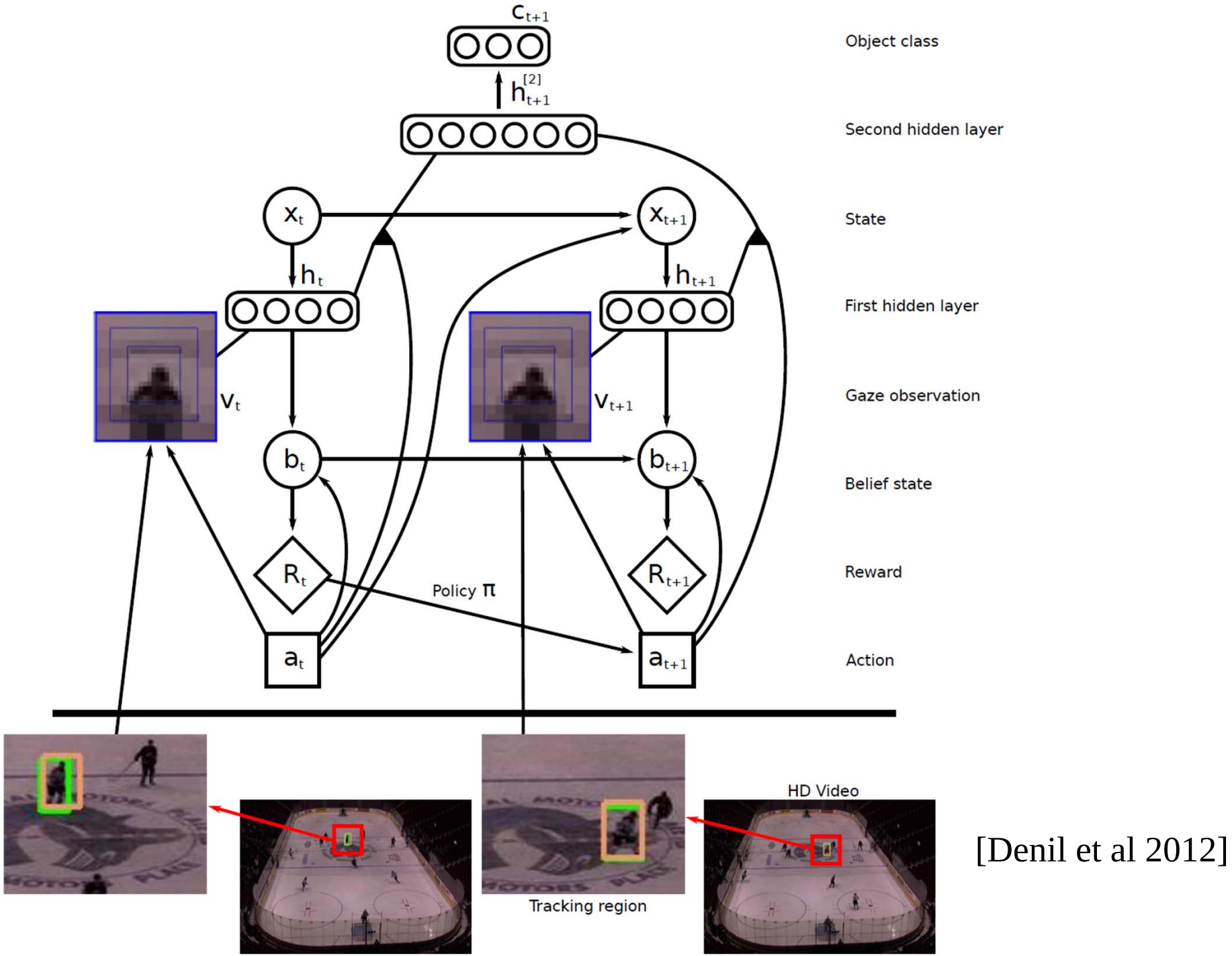
Google’s neural net learns just by watching youtube videos



Place cells in the hippocampus







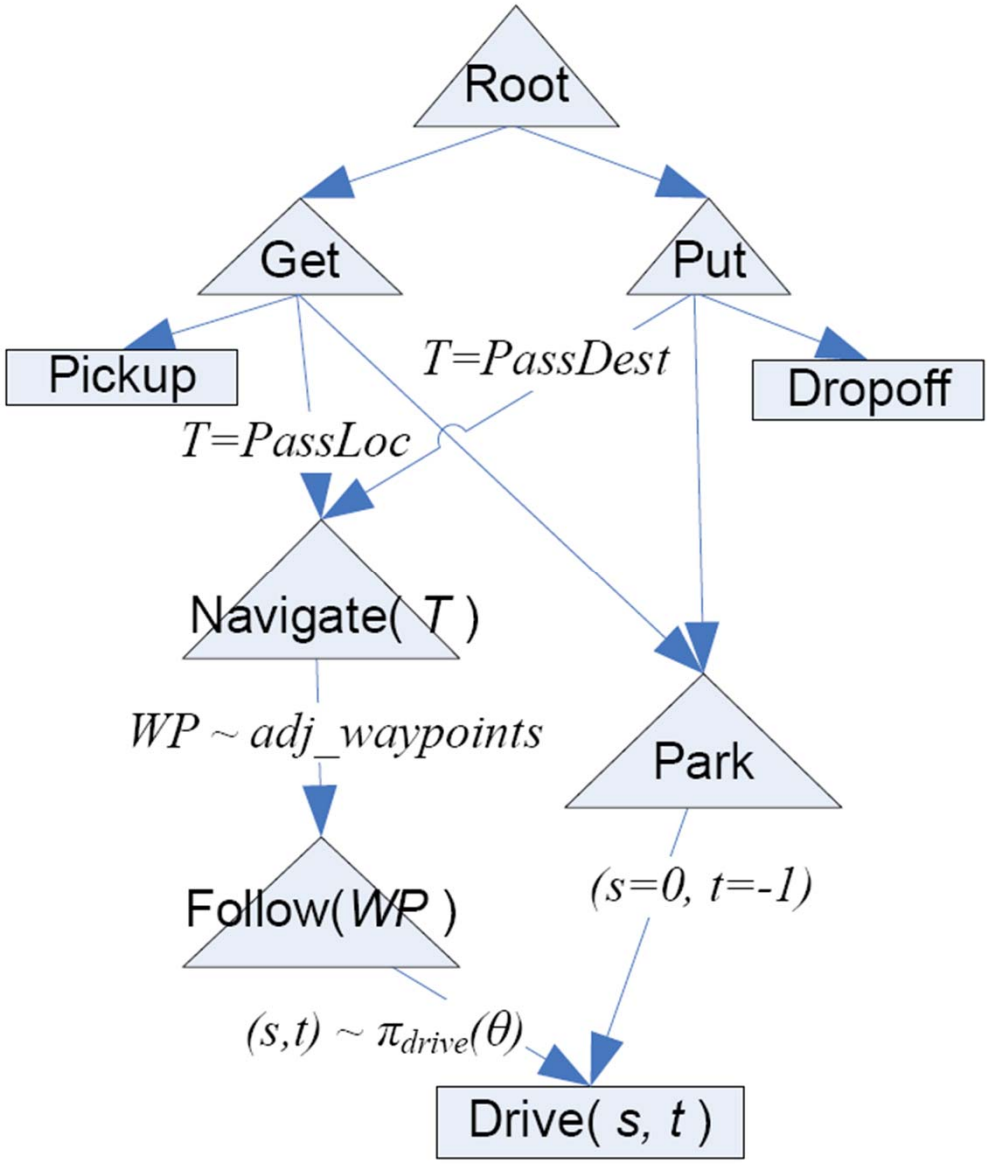
[Denil et al 2012]

Hierarchical reinforcement learning

High-level model-based learning for deciding when to navigate, park, pickup and dropoff passengers.

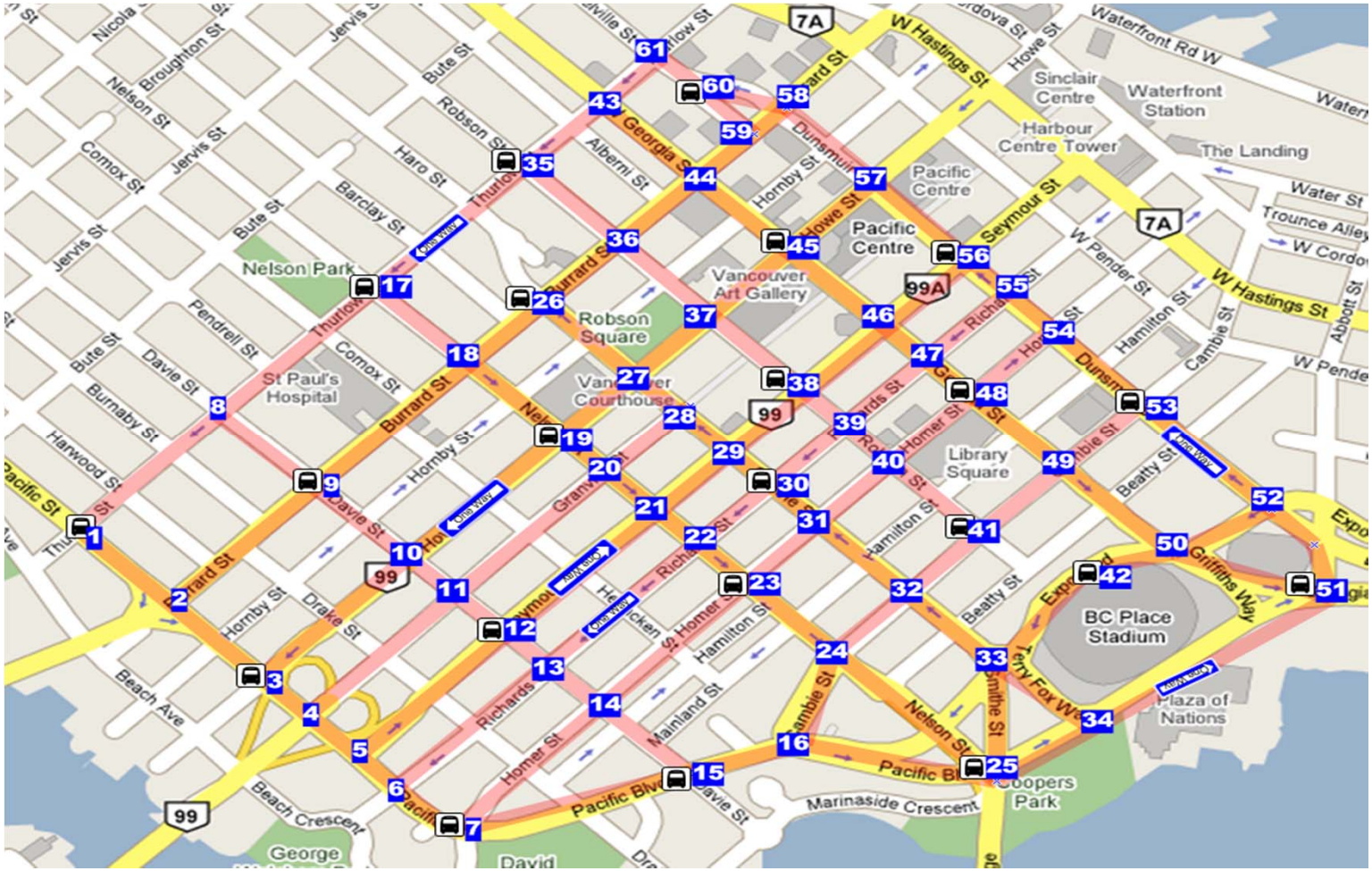
Mid-level active path learning for navigating a topological map.

Low-level active policy optimizer to learn control of continuous non-linear vehicle dynamics.



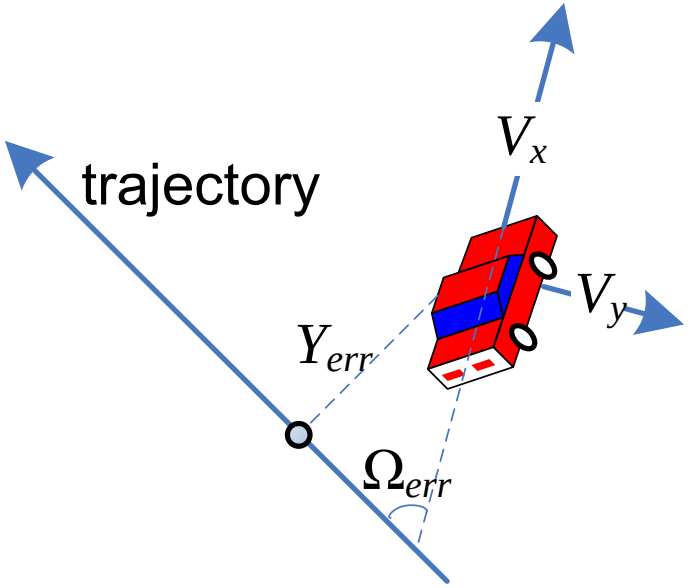
Active Path Finding in Middle Level

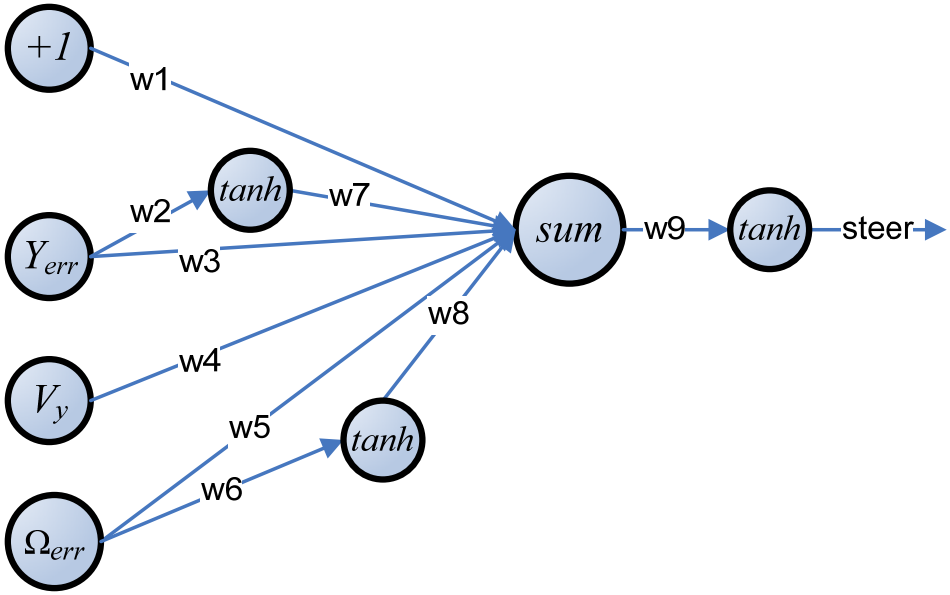
Navigate policy generates sequence of waypoints on a topological map to navigate from a location to a destination.

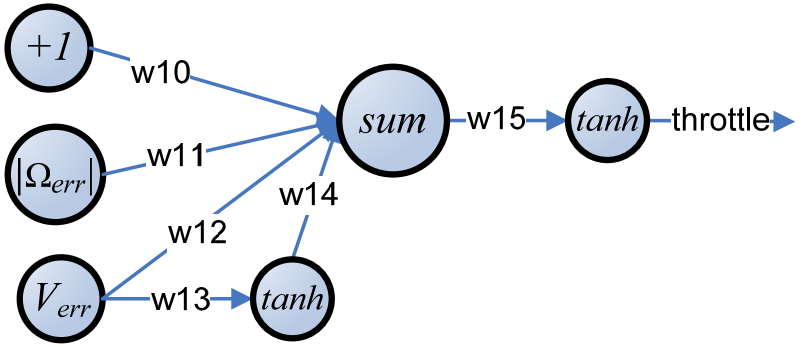


Low-Level: Trajectory following

TORCS: 3D game engine that implements complex vehicle dynamics complete with manual and automatic transmission, engine, clutch, tire, and suspension models.

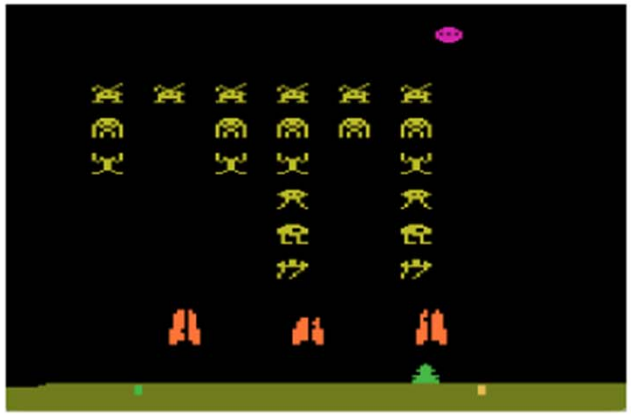


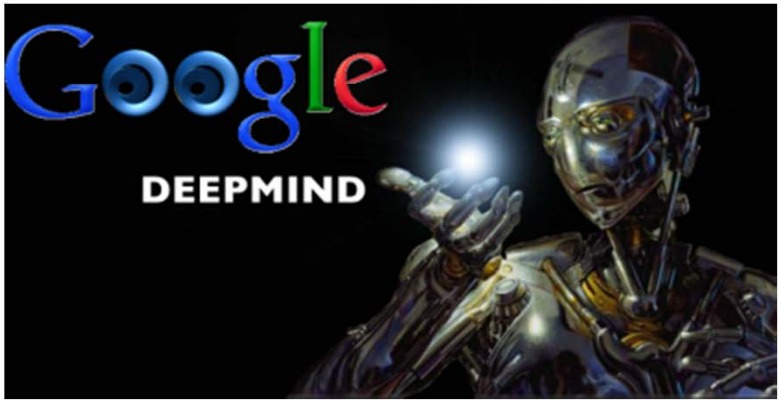




Bayesian optimization was used to find the neural net low-level policy and the value function for the upper levels

environment $\mathcal{E}$, in this case the Atari emulator.





actions, $\mathcal{A} = \{1, \ldots, K\}$

image $x_t \in \mathbb{R}^d$

$$s_t = x_1, a_1, x_2, \ldots, a_{t-1}, x_t$$

$$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$$

$$Q^*(s, a) = \max_\pi \mathbb{E}\left[R_t | s_t = s, a_t = a, \pi\right]$$

Deepmind approach

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q^*(s',a') \middle| s,a \right]$$

$$L_i(\theta_i) = \mathbb{E}_{s,a \sim \rho(\cdot)} \left[(y_i - Q(s,a;\theta_i))^2 \right]$$

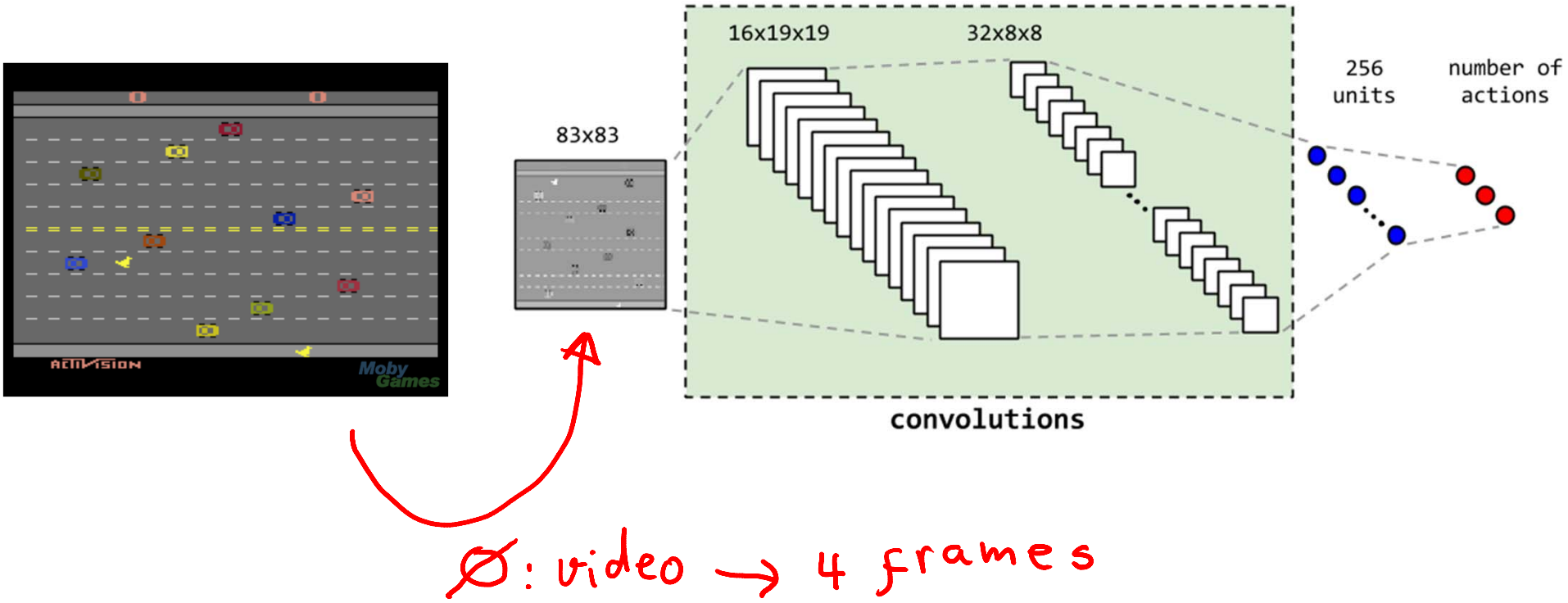
$$y_i = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) \middle| s,a \right]$$

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a \sim \rho(\cdot); s' \sim \mathcal{E}} \left[\left(r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i) \right) \nabla_{\theta_i} Q(s,a;\theta_i) \right]$$

Deepmind approach

agent's experiences $e_t = (s_t, a_t, r_t, s_{t+1})$

$$\mathcal{D} = e_1, \ldots, e_N$$

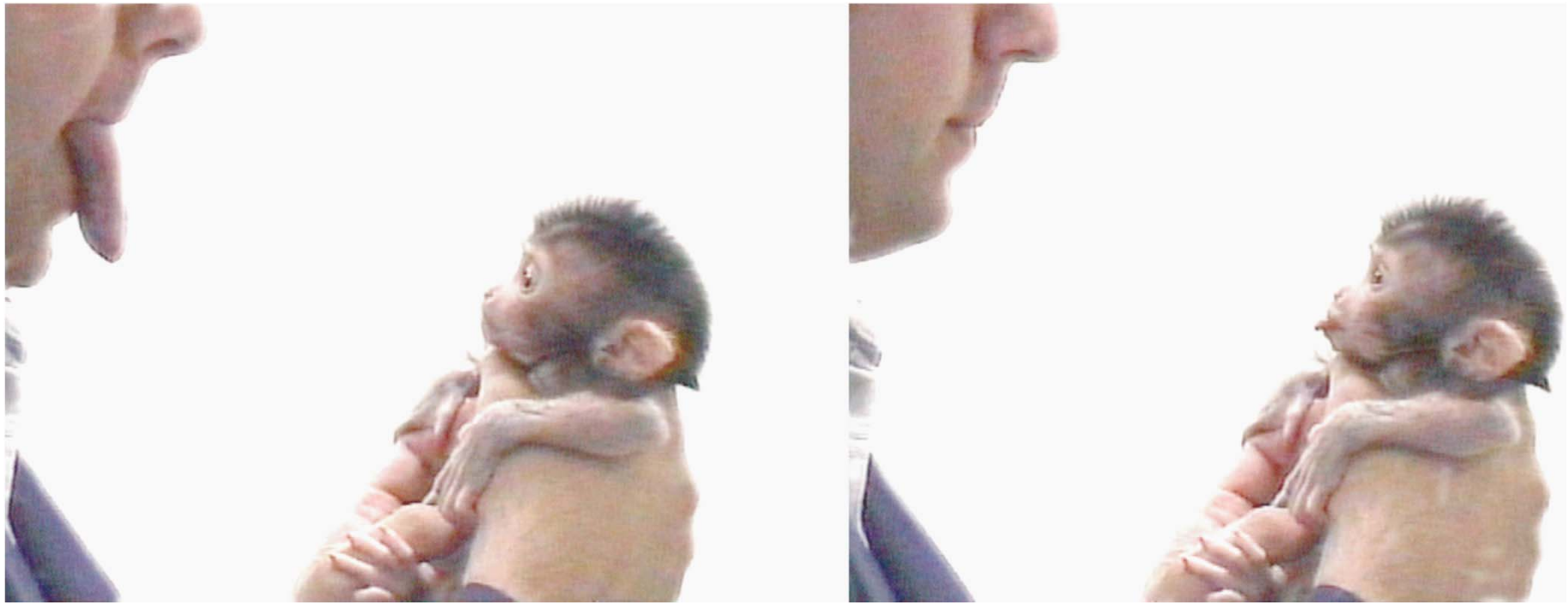


Deepmind approach

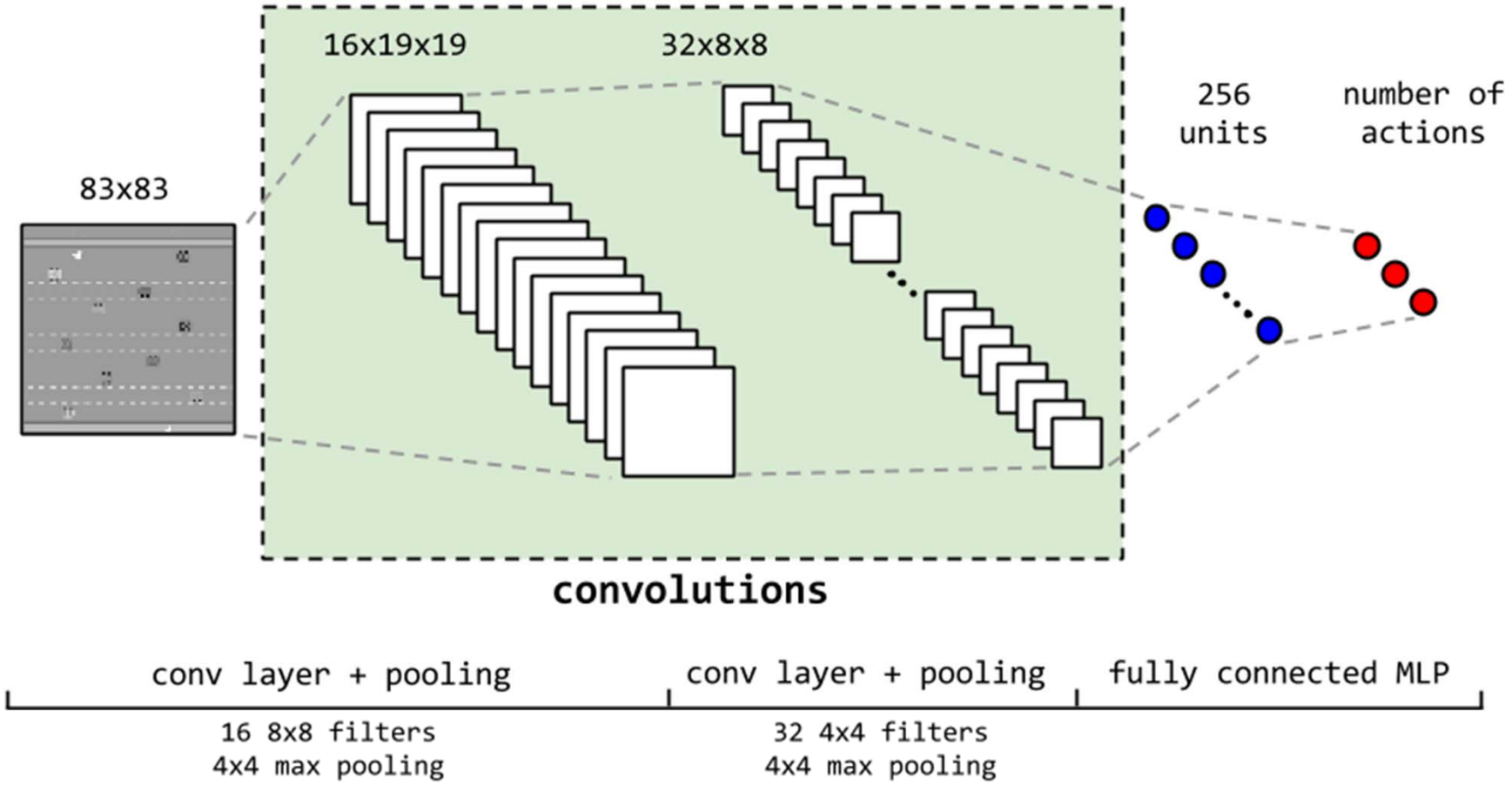
Algorithm 1 Deep Q-learning with Experience Replay

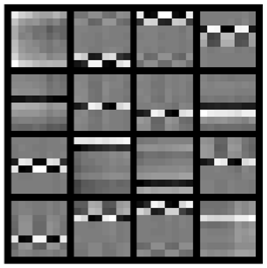
Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

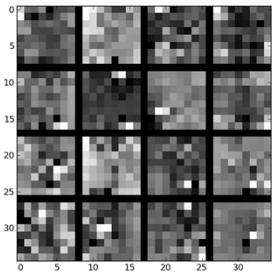
Imitation learning & mirror neurons



Imitation learning for Atari

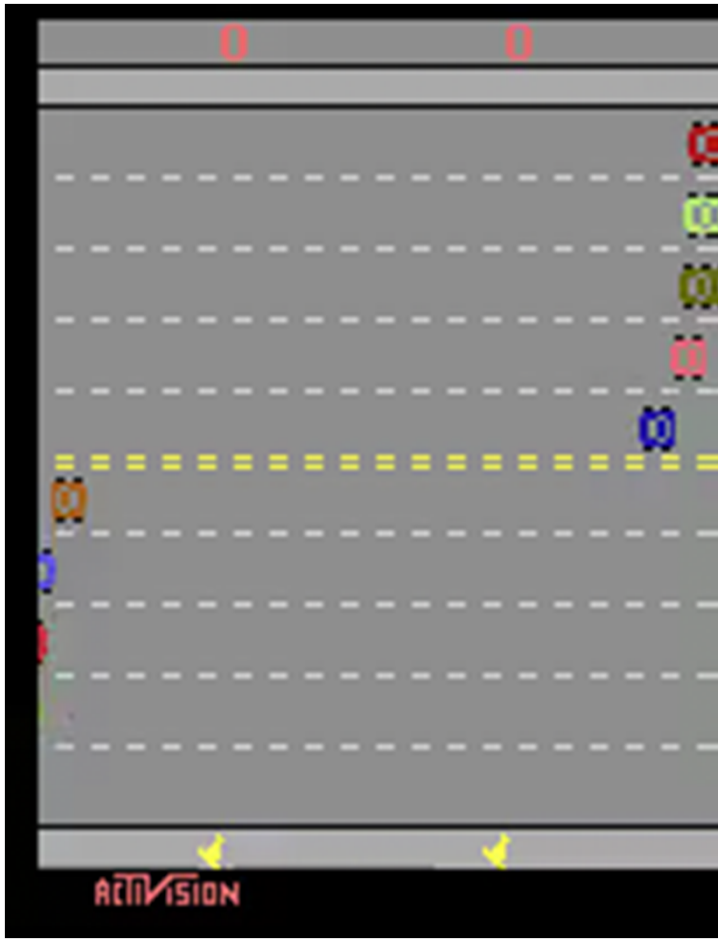






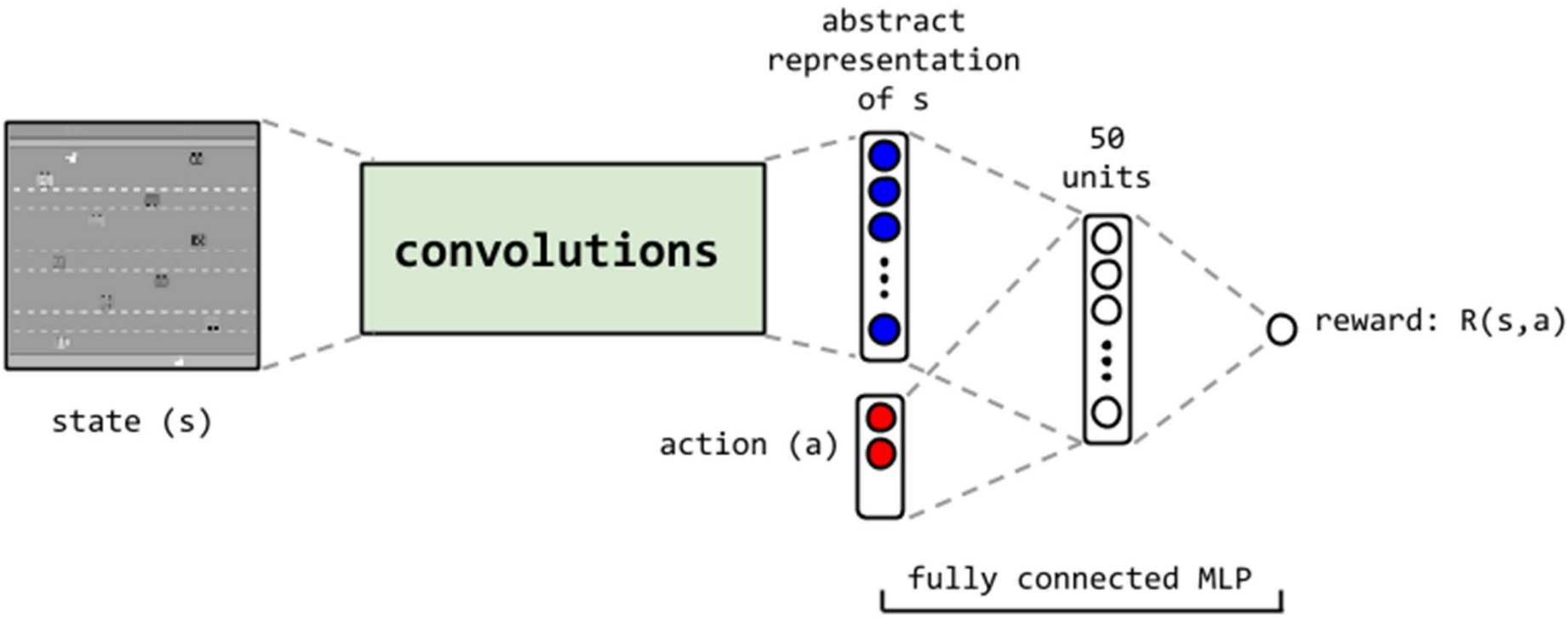
[Dejan, Miroslav, 2014]

Imitation learning for Atari



[Dejan Markovikj, Miroslav Bogdanovic, Misha Denil, NdF 2014]

Inverse RL
…or teaching deepmind how to play Atari




Thank you
Next lecture: scalable learning