Reinforcement learning

Episode 3

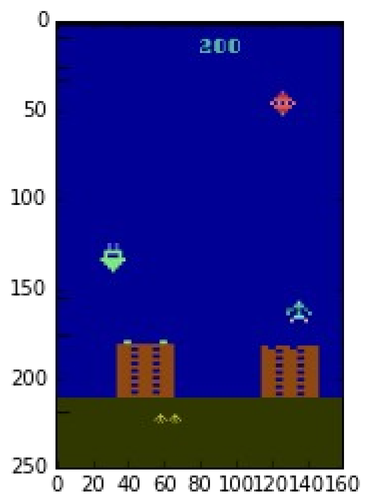
Approximate & deep RL

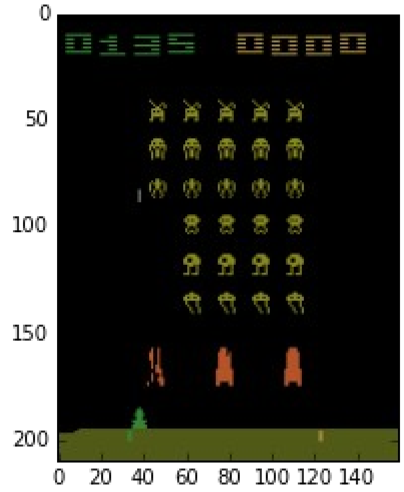
Yandex Data Factory

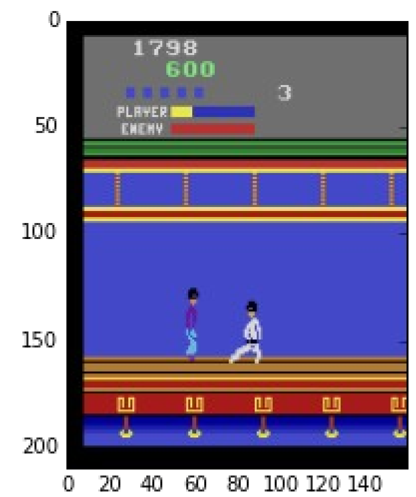
LAMBDA

British Hedgehog Preservation Society

~~Reality~~ check: videogames

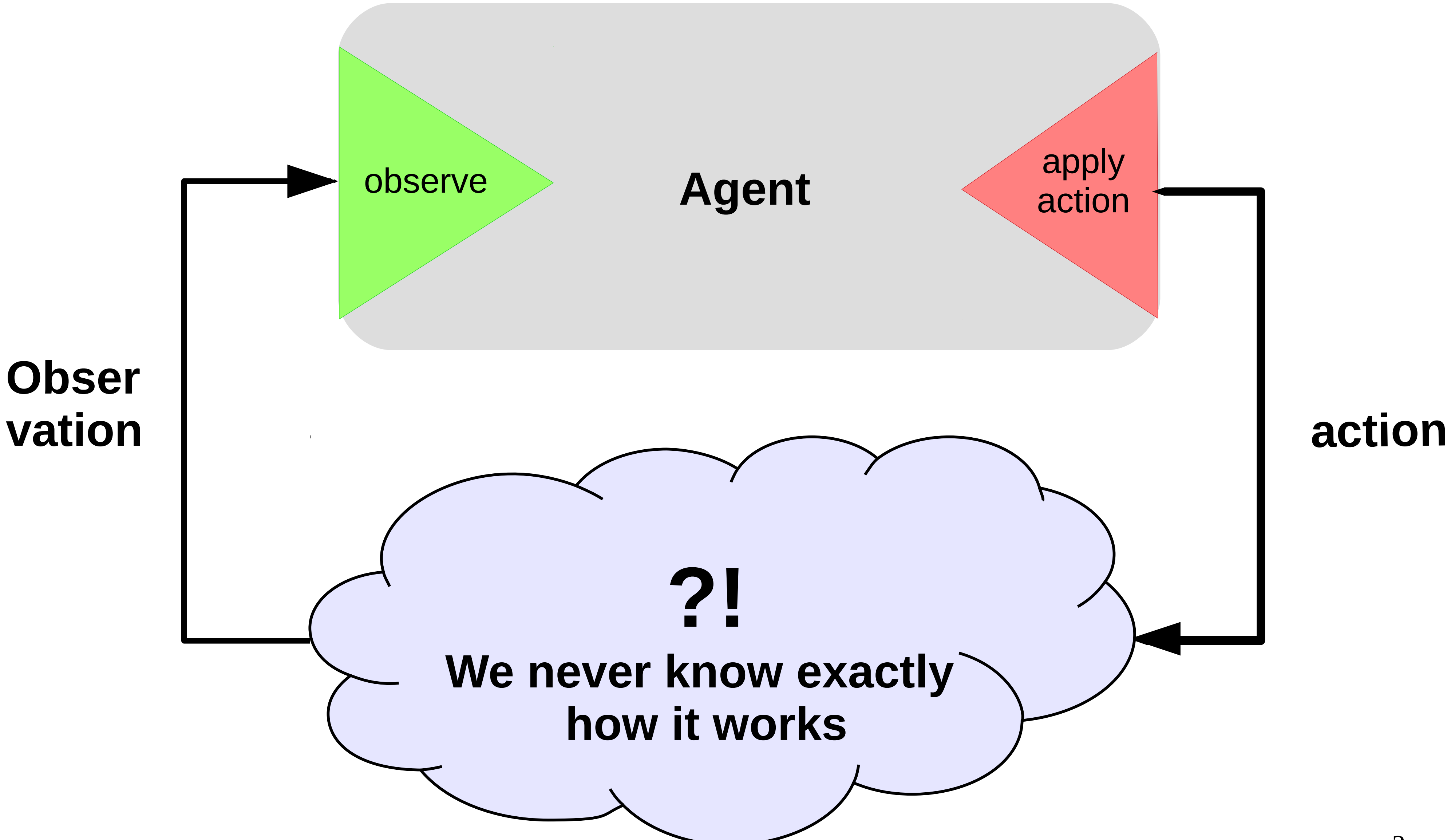




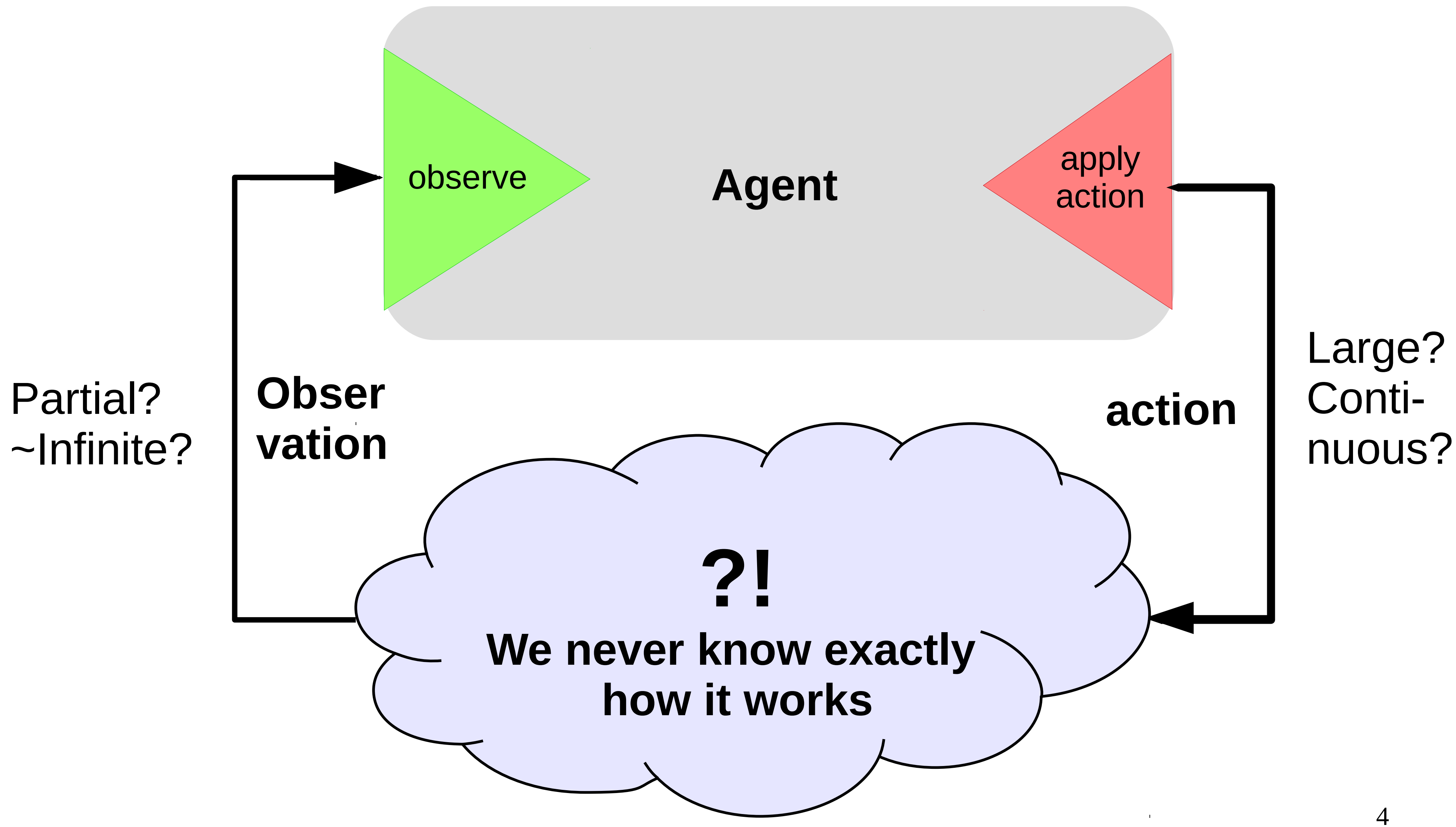


- **Trivia:** What are the states and actions?

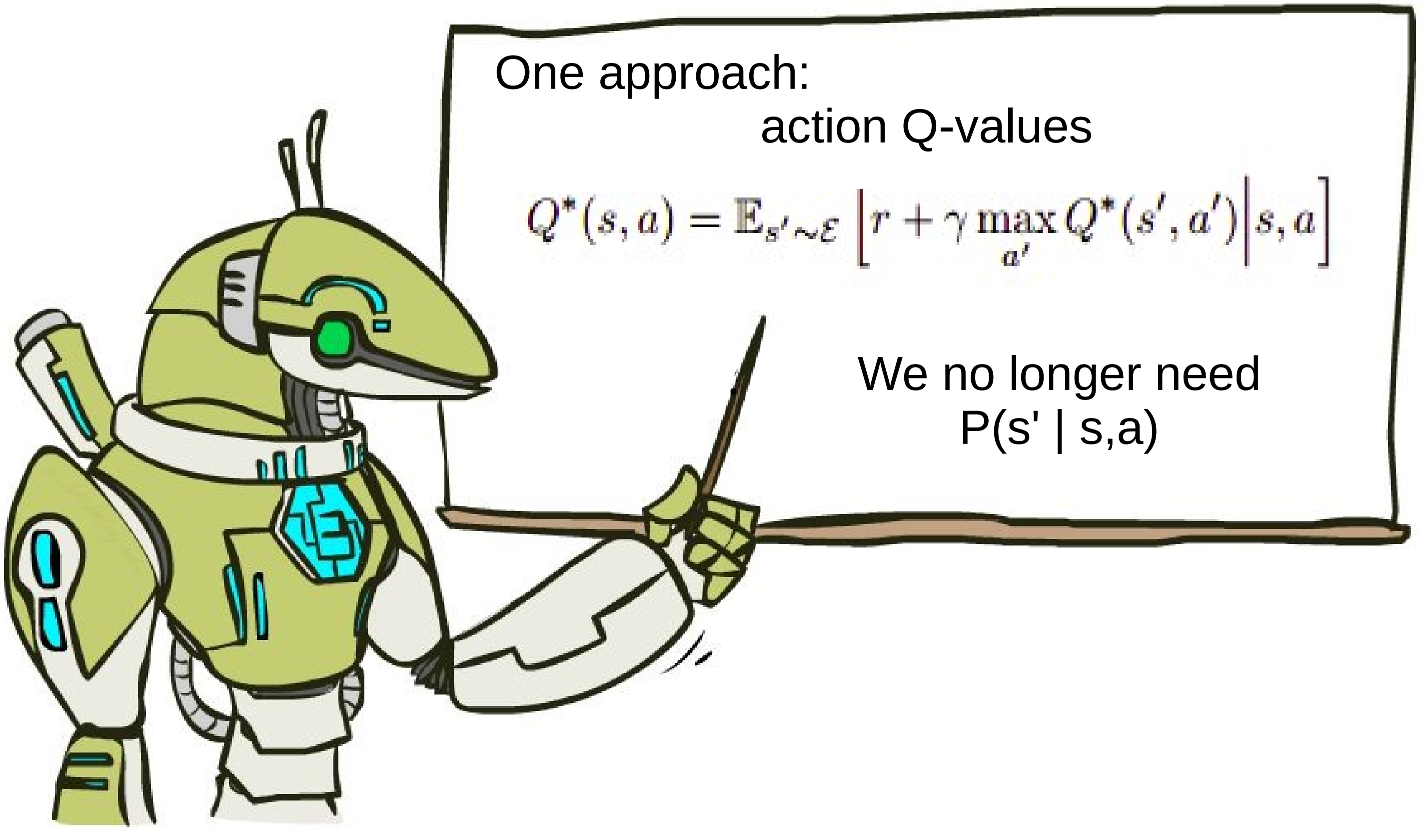
Real world


observe
Agent
apply
action
Obser
vation
action
?!
We never know exactly
how it works

Real world


observe
Agent
apply
action
Partial?
~Infinite?
Obser
vation
action
Large?
Conti-
nuous?
?!
We never know exactly
how it works

Recap: Q-learning



$$argmin_Q \left(Q(s_t, a_t) - [r_t + \gamma \cdot max_{a'} Q(s_{t+1}, a')] \right)^2$$

$$\pi(s) : argmax_a Q(s, a)$$

Problem:

State space is usually large,

sometimes continuous.

And so is action space;

However, states do have a structure, similar states have similar action outcomes.

From tables to approximations

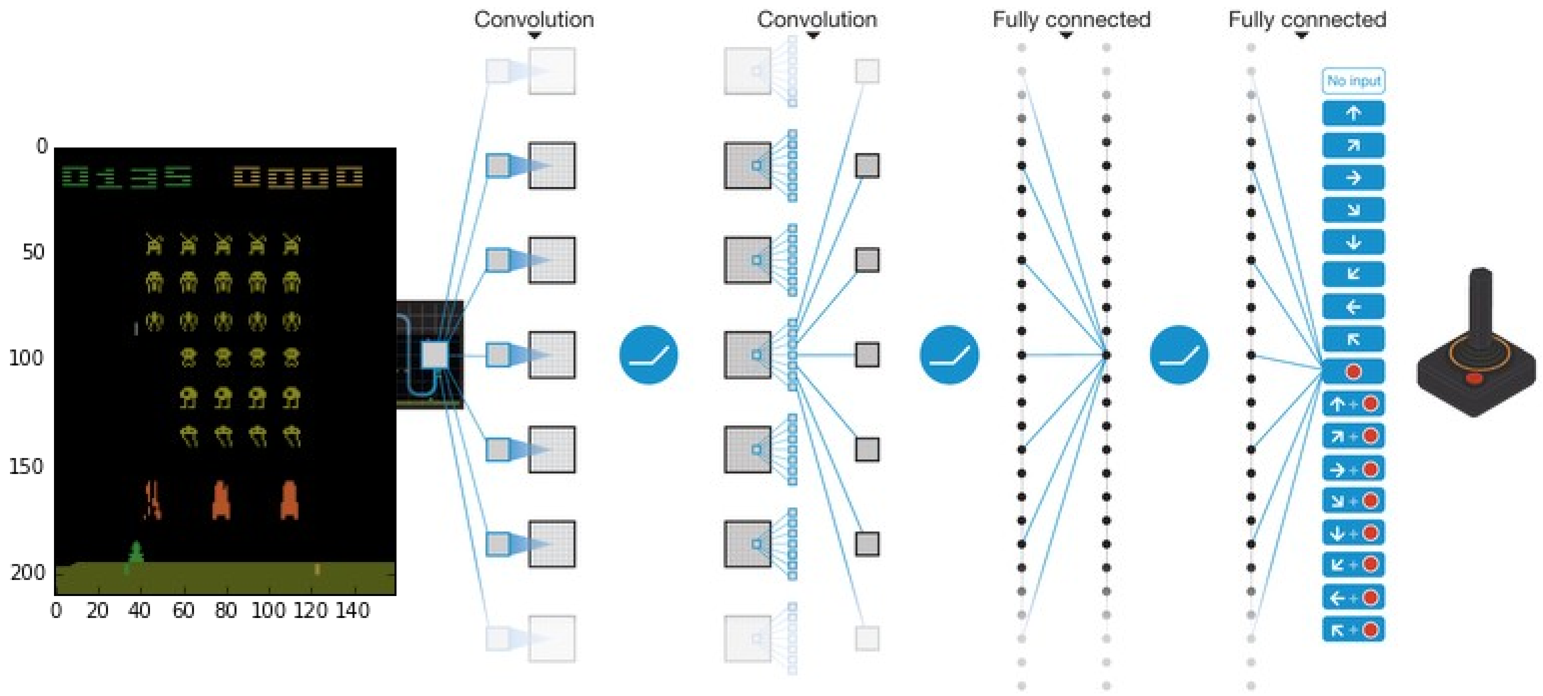
- Before:
  - For all states, for all actions, remember Q(s,a)

- Now:
  - Approximate Q(s,a) with some function
  - e.g. linear model over state features

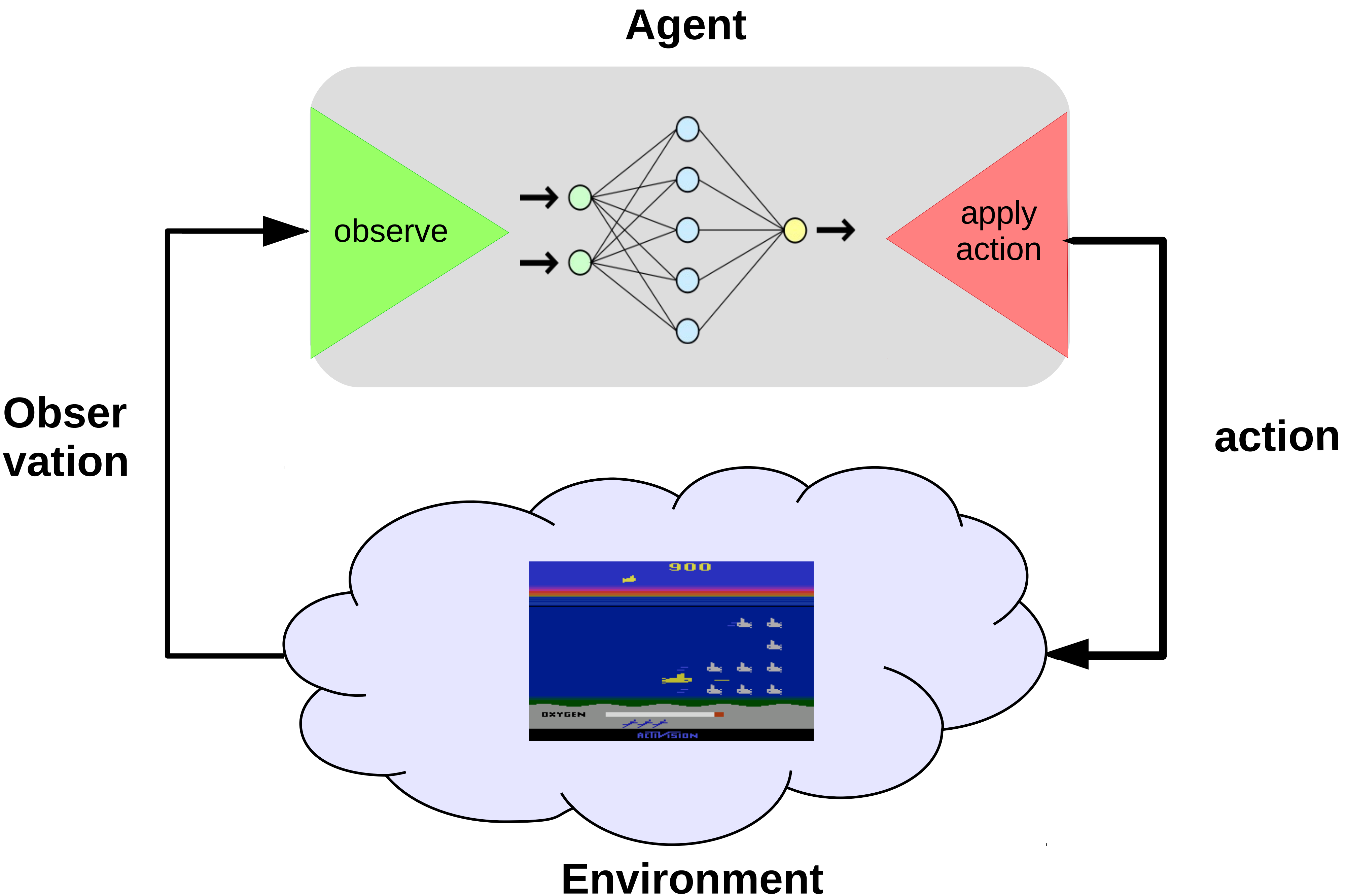
$$argmin_{w,b}\left(Q(s_t,a_t)-[r_t+\gamma\cdot max_{a'}Q(s_{t+1},a')]\right)^2$$

Trivia: should we use linear regression or logistic regression?

Deep learning approach: DQN

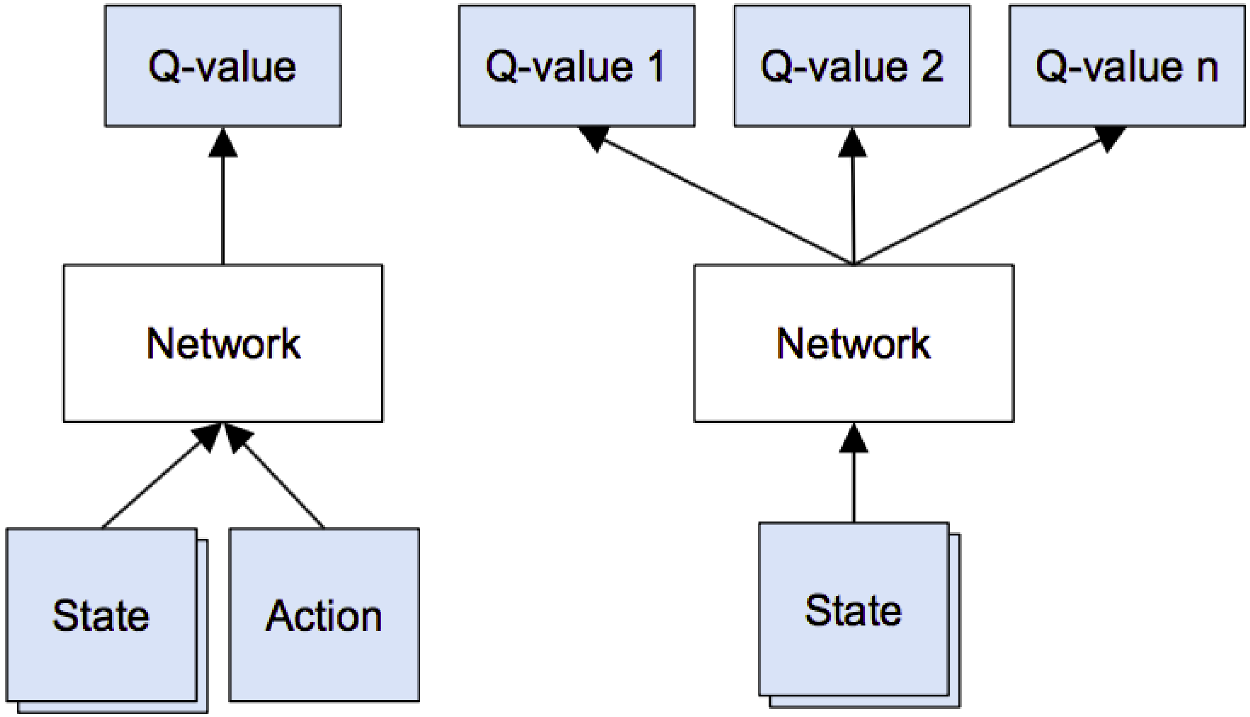


MDP again



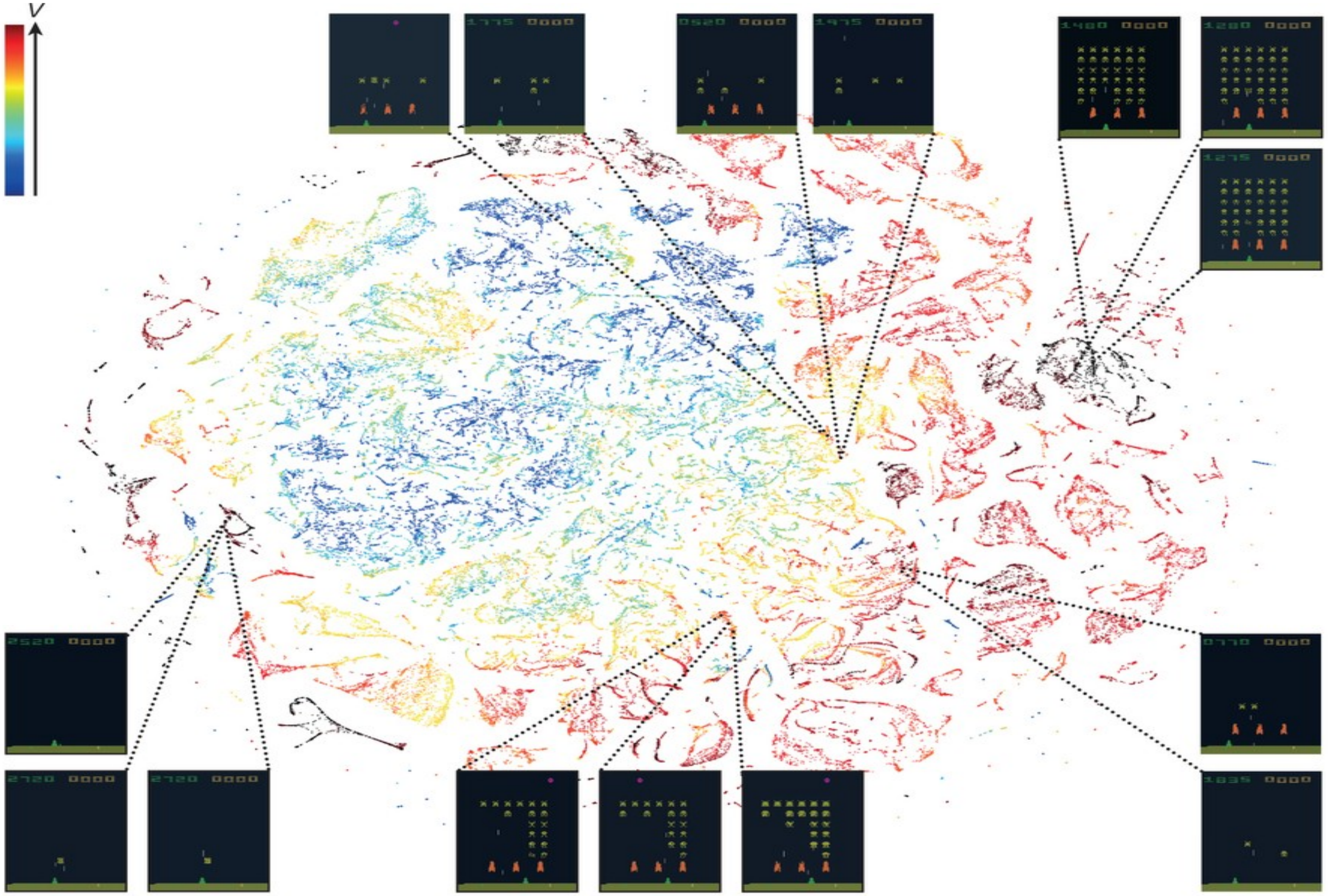
Agent
observe
apply
action
Obser
vation
action
900
OXYGEN
ACTIVISION
Environment

Deep learning approach: DQN

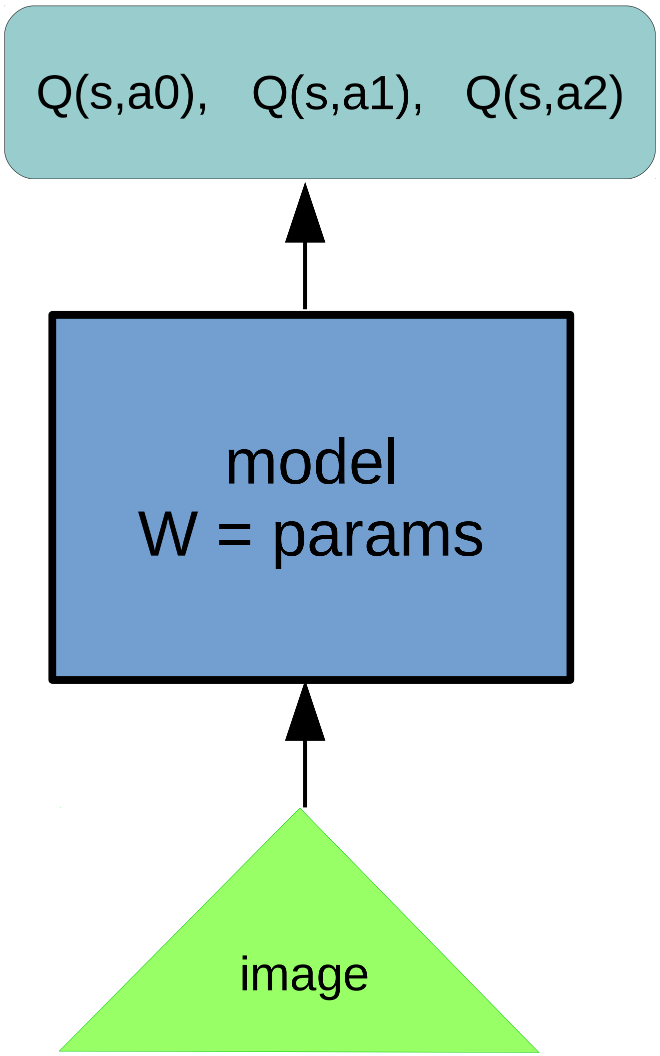


$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim \mathrm{U}(D)} \left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i) \right)^2 \right]$$

Deep learning approach: DQN



Approximate Q-learning



Q-values:

$$\hat{Q}(s_t, a_t) = r + \gamma \cdot argmax_a{}' \hat{Q}(s_{t+1}, a')$$

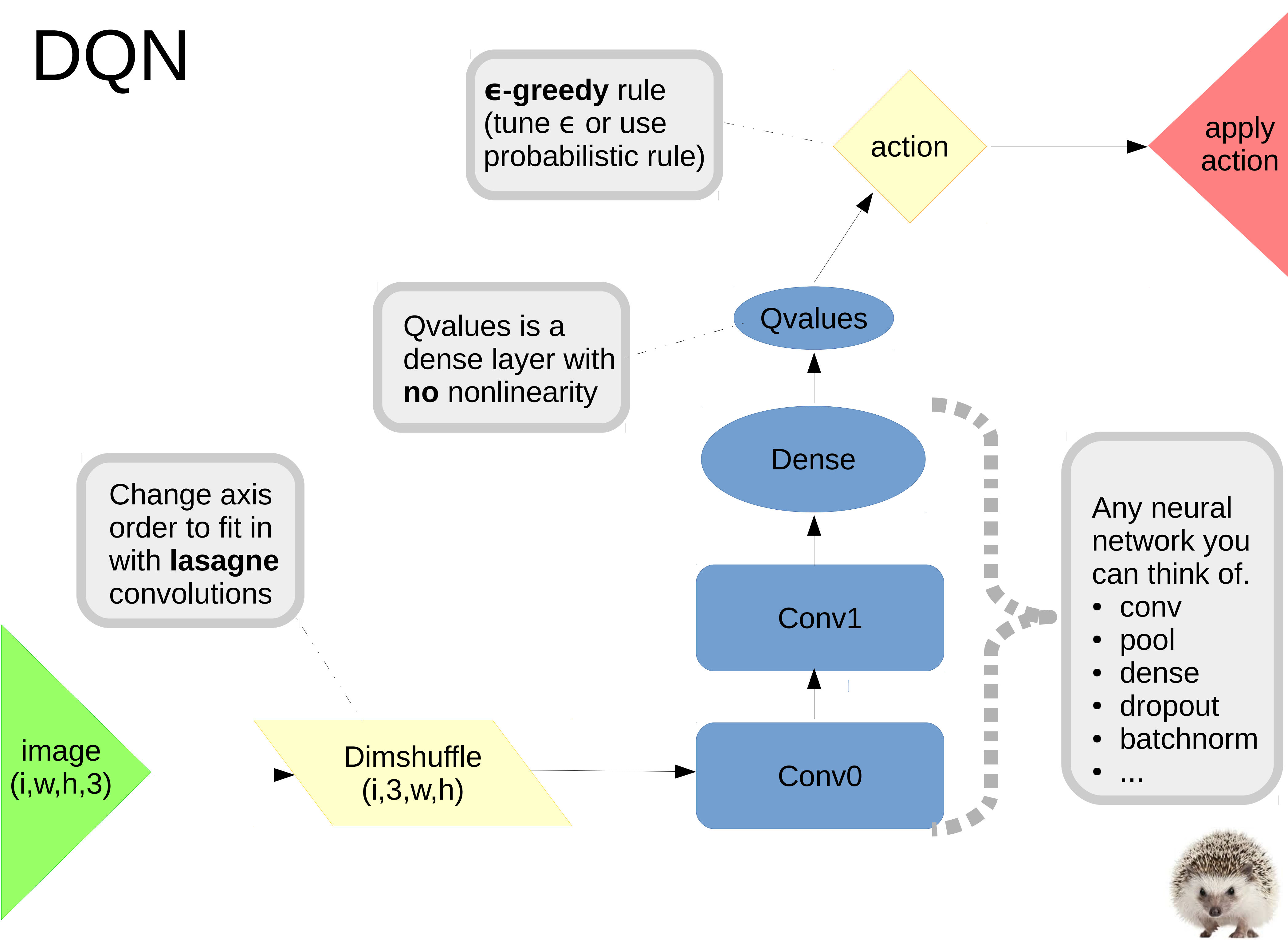
Objective:

$$L = (Q(s_t, a_t) - r + \gamma \cdot argmax_a{}' Q(s_{t+1}, a'))^2$$

Gradient step:

$$w_{t+1} = w_t - \alpha \cdot \frac{\delta L}{\delta w}$$

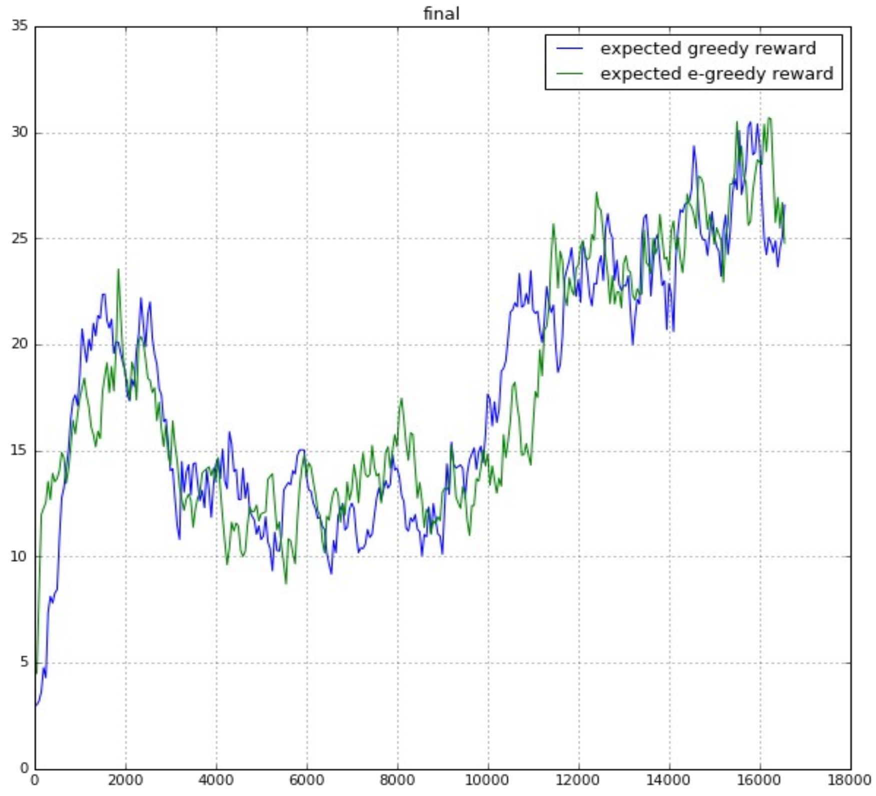
DQN



ϵ-greedy rule
(tune ϵ or use
probabilistic rule)
action
apply
action
Qvalues is a
dense layer with
no nonlinearity
Qvalues
Dense
Conv1
Conv0
Change axis
order to fit in
with lasagne
convolutions
image
(i,w,h,3)
Dimshuffle
(i,3,w,h)
Any neural
network you
can think of.
• conv
• pool
• dense
• dropout
• batchnorm
• ...

Approximate Q-learning

- Training samples are **not “i.i.d”**,
- Model forgets parts of environment it haven't visited for some time,
- Fallbacks on the learning curve
- **Any ideas?**



Decorrelating

Experience replay

- Maintain a large pool of (s,a,r,s') tuples from prior MDP sessions.
- Sample random batch from the pool each time when training NN
  - Or use a prioritized sampling strategy to emphasize most important samples

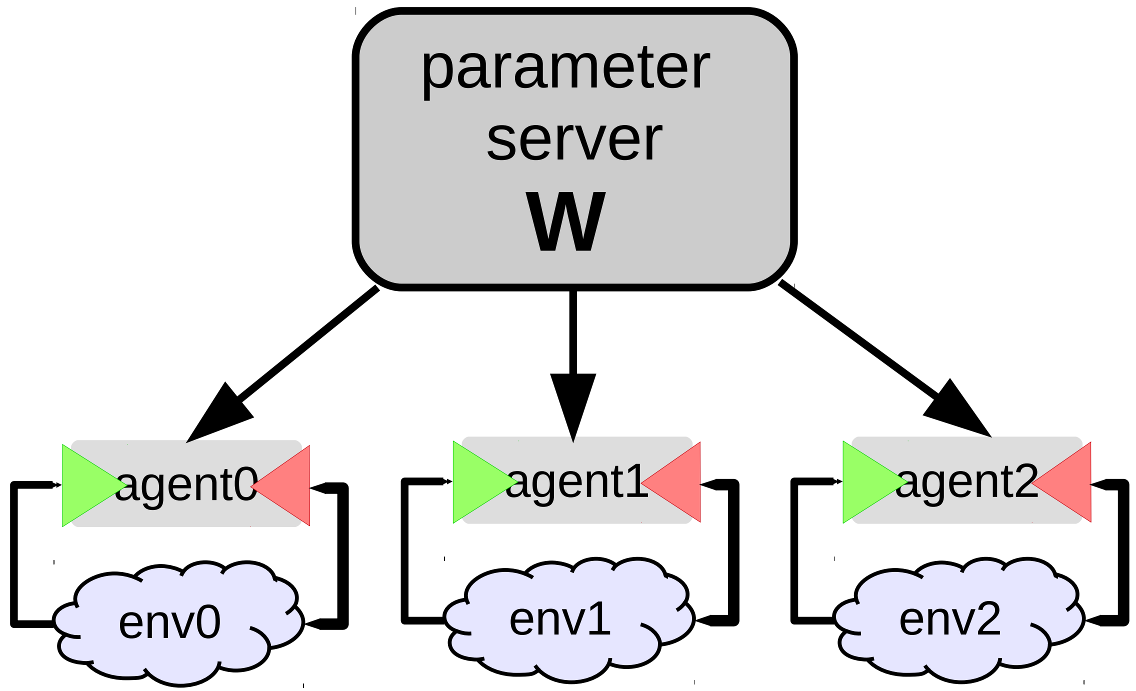
Target networks

- Obtain “Qreference(s,a)” term from an older neural network snapshot.
  - Alternatively, maintain an exponential moving average of weights

Multiple agent trick

Idea: Throw in several agents with shared **W**.

- Chances are, they will be exploring different parts of the environment,
- More stable training,
- Requires a lot of interaction,
- Alternative to experience replay.



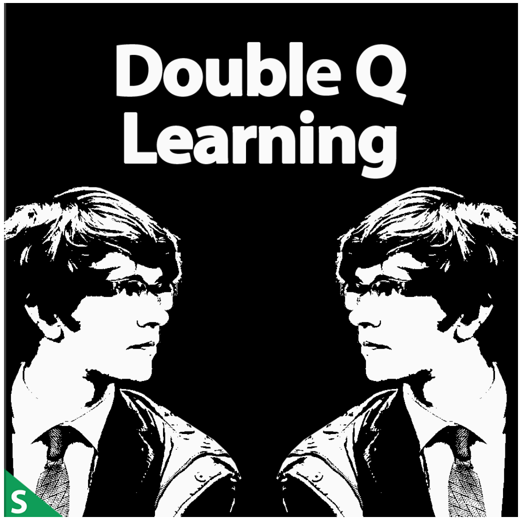
More decorrelating

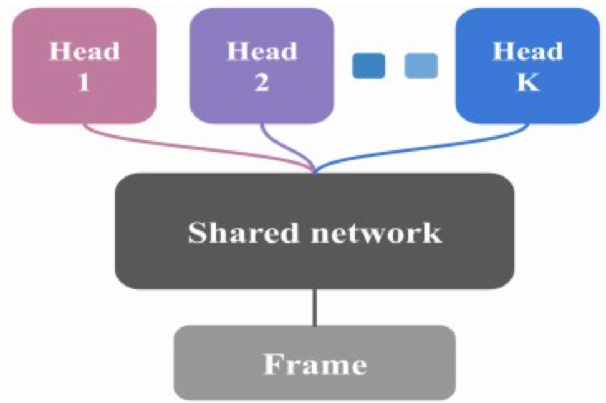
Double Q-learning

- Maintain two Q-networks
- use one to pick best action and the other to evaluate it's Q-value

Bootstrap DQN

- Maintain several “heads”, top layers of NN responsible for Qvalues prediction.
- At the beginning of new game session, choose one of the “heads” at random.
- This head decides what action to take during current session.
- All other heads are trained on that session without taking any real actions





Problem:

Most practical cases are partially observable:

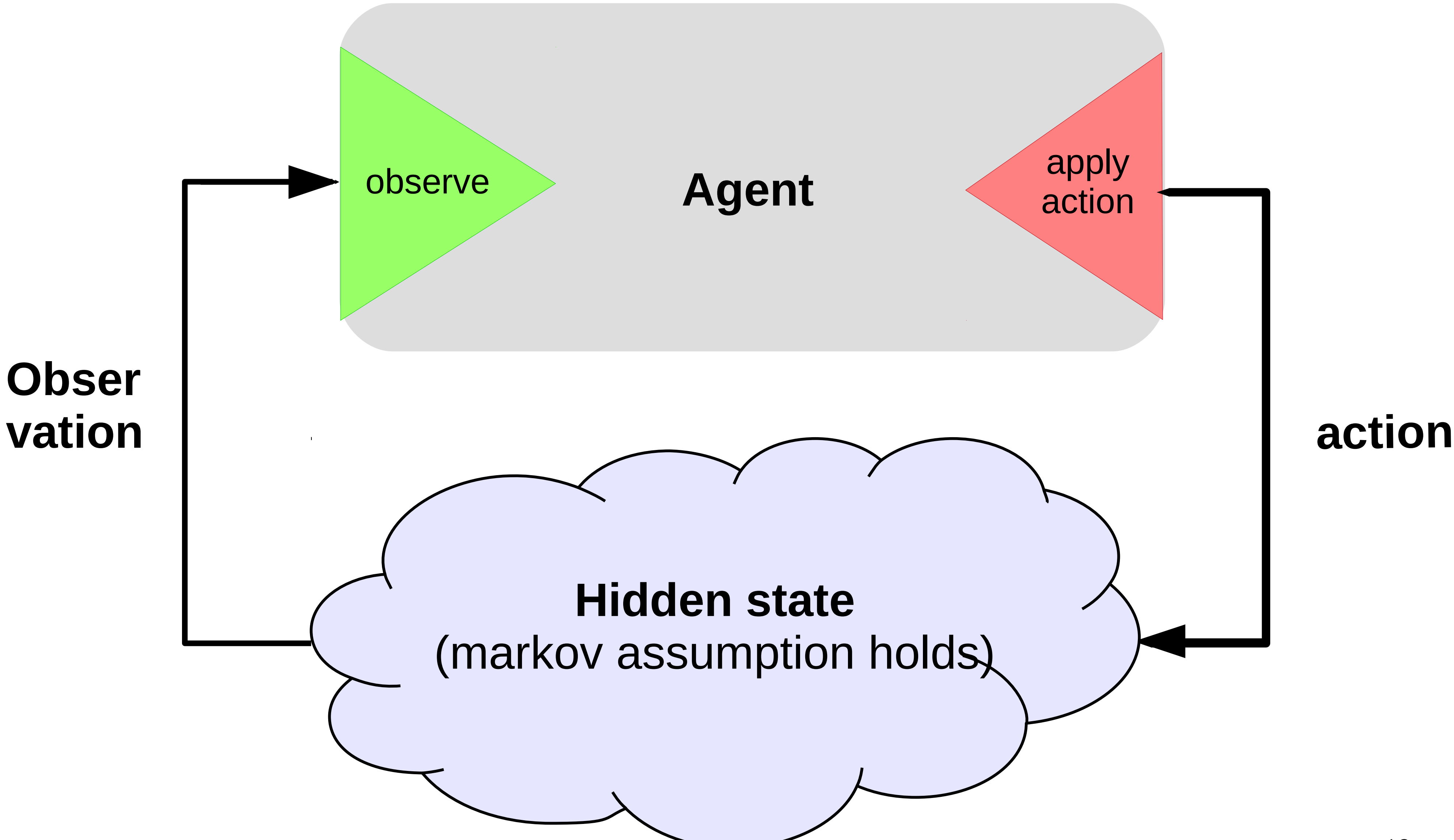
Agent observation does not hold all information about process state (e.g. human field of view).

- However, we can try to infer hidden states from sequences of observations.

$$s_t \simeq m_t : P\left(m_t | o_t, m_{t-1}\right)$$

- Intuitively that's agent memory state.

Partially observable MDP



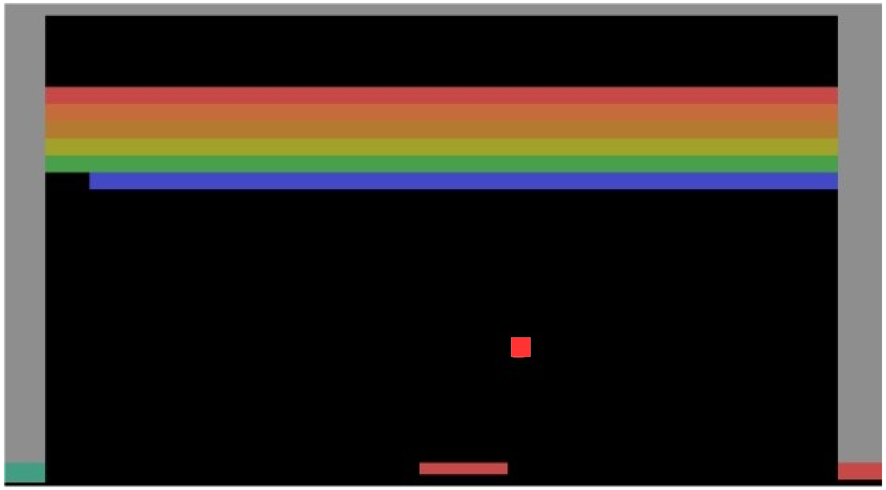
N-gram heuristic

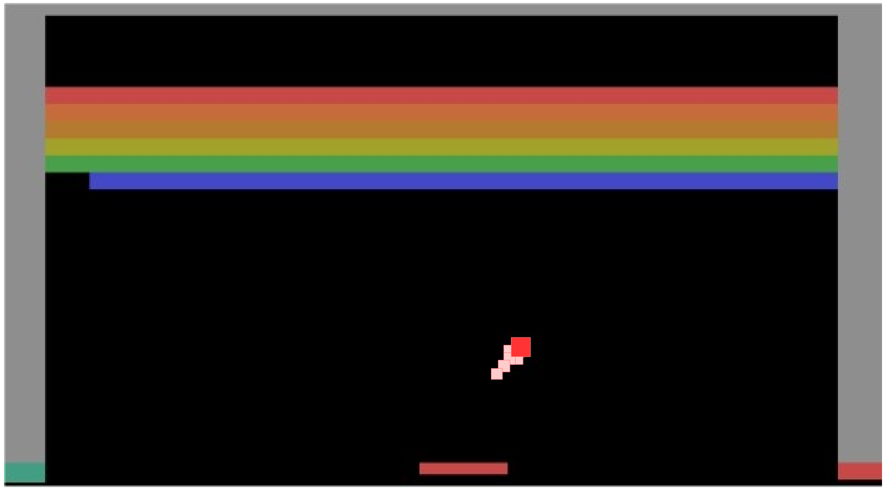
Idea:

$$s_t \neq o(s_t)$$

$$s_t \approx (o(s_{t-n}), a_{t-n}, \ldots, o(s_{t-1}), a_{t-1}, o(s_t))$$

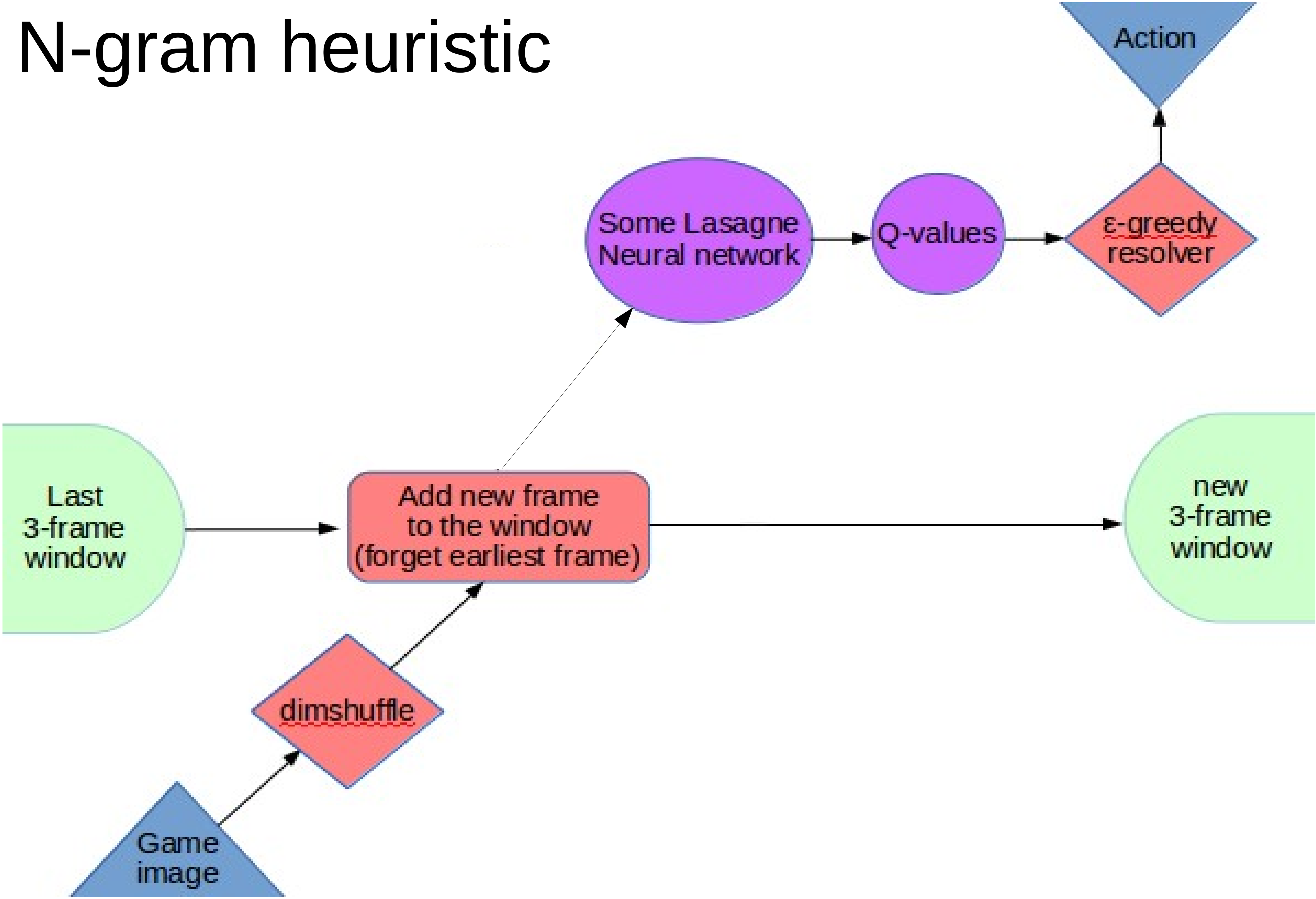
e.g. ball movement in breakout



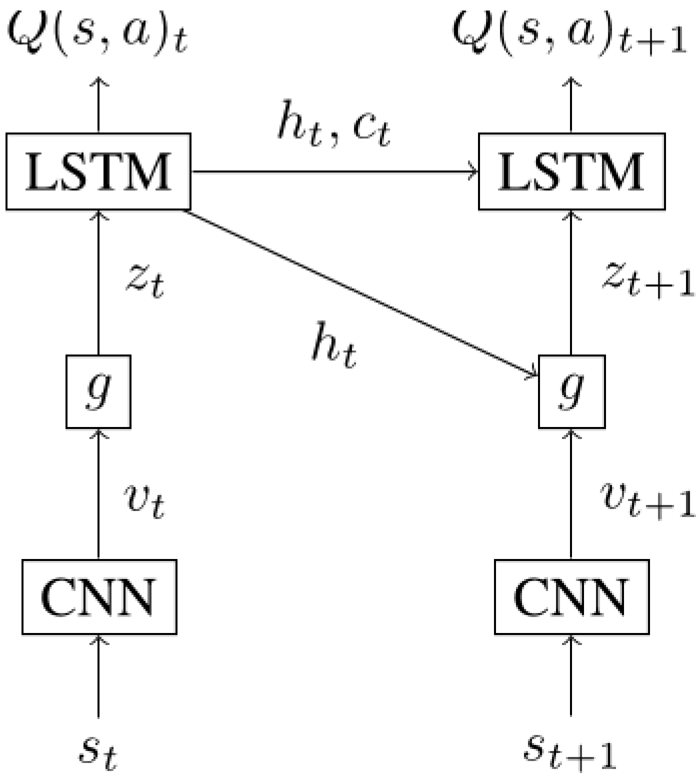


- Does ball fly up or down?
- Several frames

N-gram heuristic

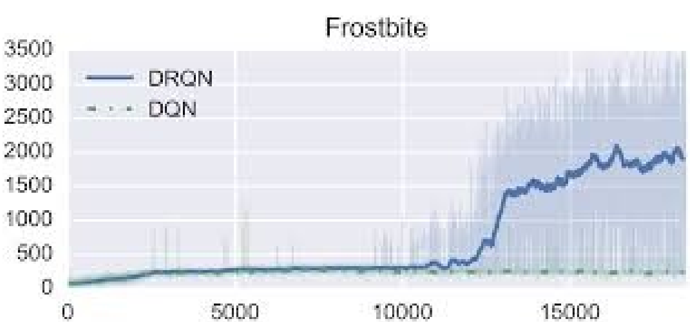


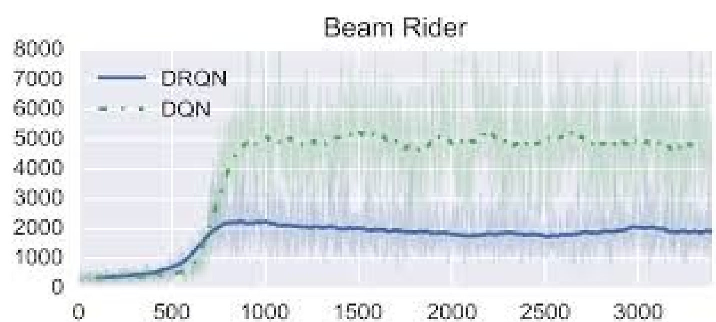
Deep Recurrent RL

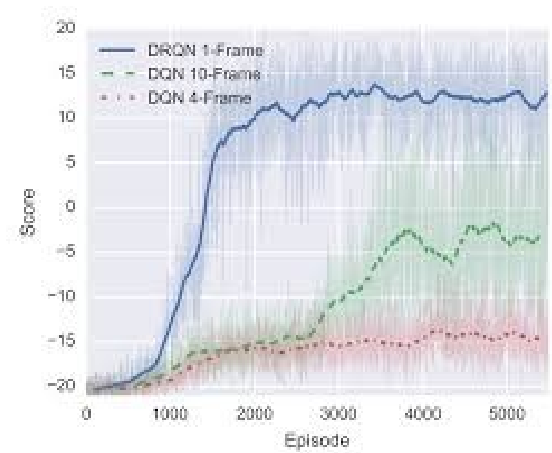


Recurrent agent memory

- Agent has his own hidden state.
- Trained via BPTT with a fixed depth
- Problem: next input depends on chosen action
- Even more autocorrelations :)

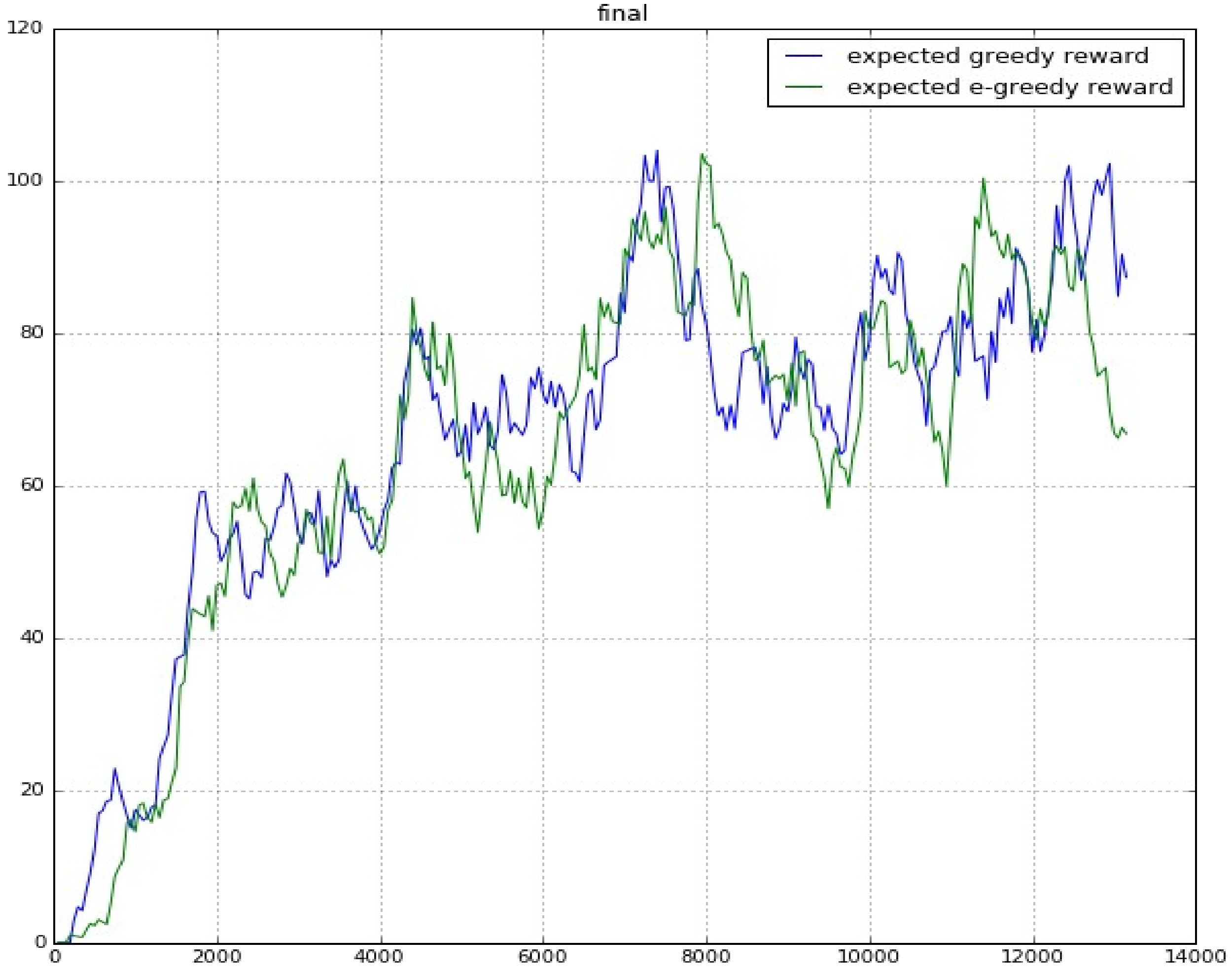






Deep Recurrent RL

Learning curves for KungFuMaster



Most important slide

RL isn't magical

- It won't learn everything in the world given any data and random architecture.
- Sparse & delayed rewards still a problem
- Less playing Atari, more real world problems
  No, doom is not a real world problem, dummy!
- Slowly getting rid of heuristics towards mathematical soundness
- Machine Intelligence revolution date TBA

Let's go play some atari!