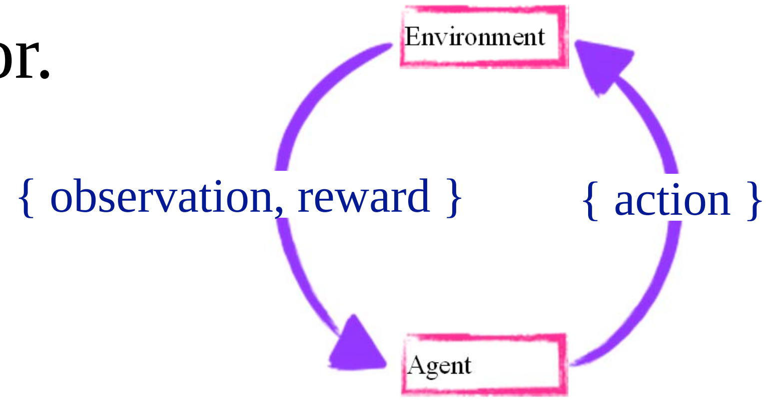
Reinforcement learning

Nando de Freitas

UNIVERSITY OF OXFORD

The Promise of Reinforcement Learning

Learning to act through trial and error.

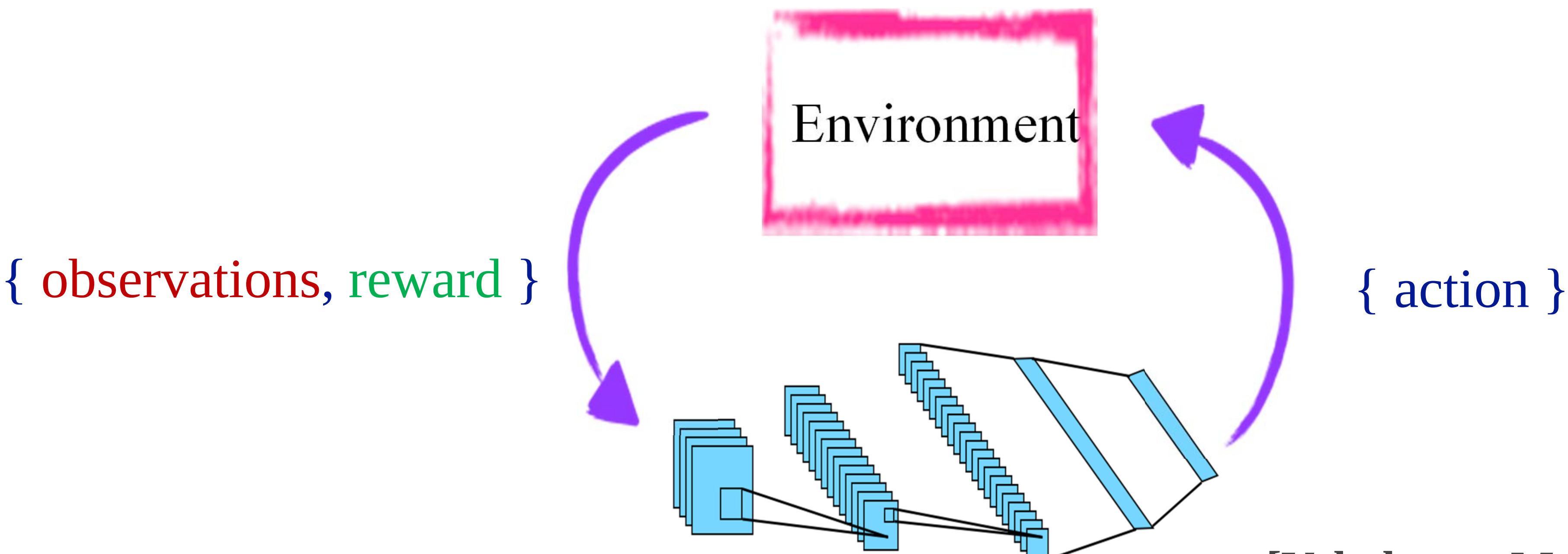


- An agent interacts with an environment and learns by maximizing a scalar reward signal.
- No models, labels, demonstrations, or any other human-provided supervision signal.
- Representation has been a challenge/missing.

[Volodymyr Mnih]

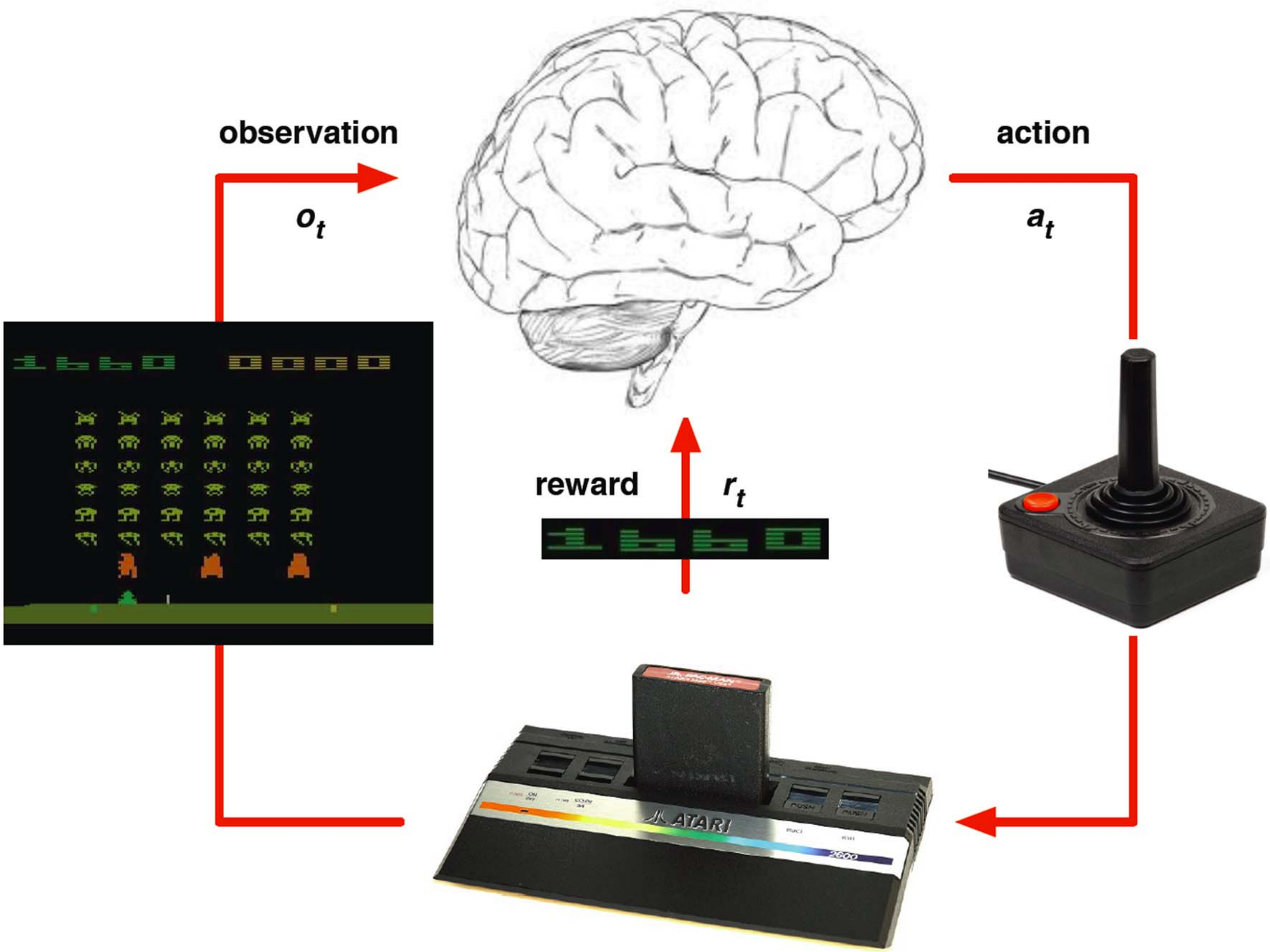
Deep Reinforcement Learning

- Combining deep neural networks with RL.
- Learn to act from high-dimensional sensory inputs.
- Is a noisy, sparse, and delayed reward signal sufficient for training deep networks? Credit assignment problem.



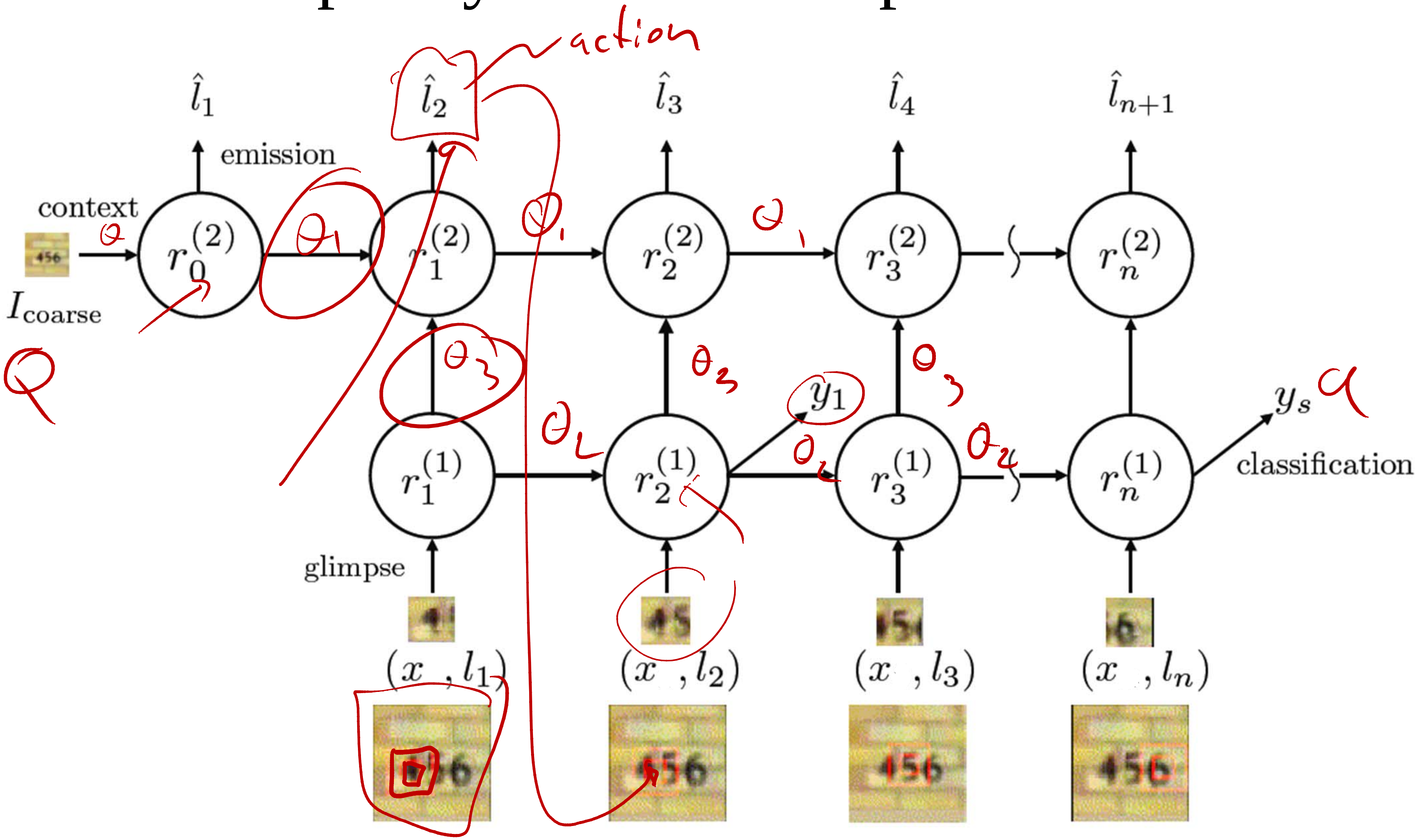
[Volodymyr Mnih]

Example: Learning to play Atari



David Silver

Direct policy search example: Attention



[Ba, Mnih, Kavuckuoglu]

Results

MNIST sequences

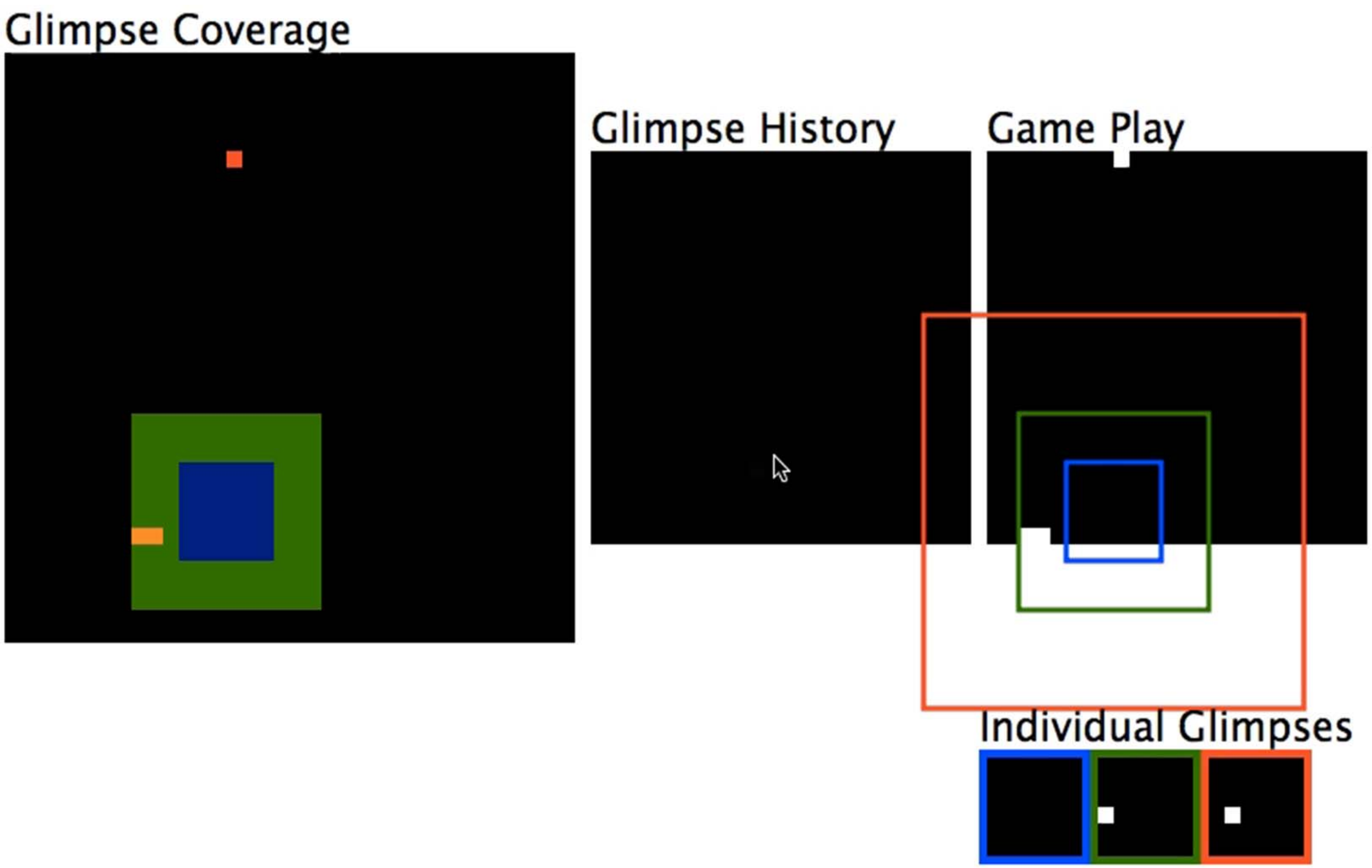
Street View House Number sequences

- The attention-based model achieves state-of-the-art accuracy on the SVHN multi-digit task - 3.9% error.
- 4 times fewer floating point operations than the best ConvNet.

[Volodymyr Mnih et al]

Attention-Based Game Agent

- Roughly the same model and training method can be used in a game-playing agent.
- The agent learns to track a ball without being told to do so.

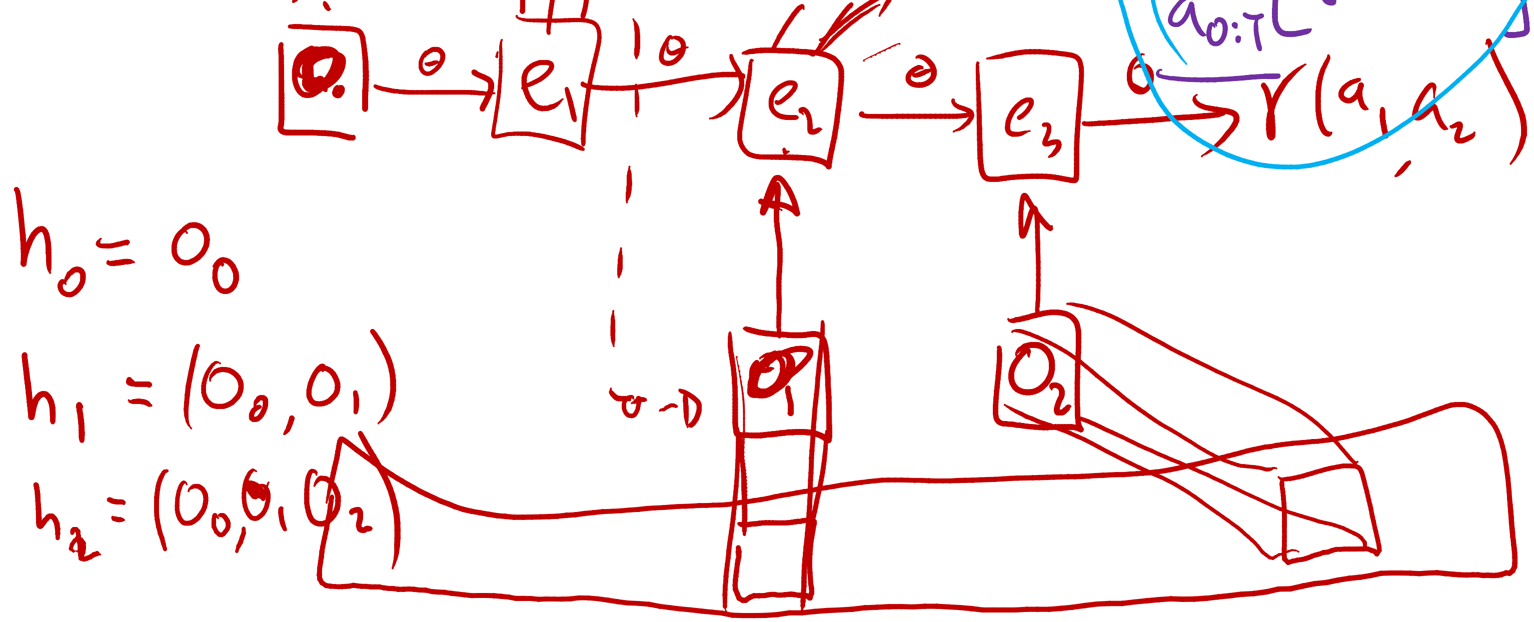


Direct policy search

$$\mathbf{h}_t = \{\mathbf{o}_0, \mathbf{o}_2, \ldots, \mathbf{o}_t\}$$

$$\pi_{\boldsymbol{\theta}}(\mathbf{a}_{0:T}|\mathbf{h}_{0:T}) = \prod_{t=0}^{T} \pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{h}_t)$$

$$J^{\pi}(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=0}^{T} r_t(\mathbf{a}_t)\right] = \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\mathbf{a}_{0:T}|\mathbf{h}_{0:T})}\left[\sum_{t=0}^{T} r_t(\mathbf{a}_t)\right]$$



Policy gradients using backprop

$$\nabla_{\boldsymbol{\theta}} J^{\pi}(\boldsymbol{\theta}) = \sum_{\mathbf{a}_{0:T}} \left[\sum_{t=0}^{T} r_t(\mathbf{a}_t) \right] \nabla \pi_{\boldsymbol{\theta}}(\mathbf{a}_{0:T} | \mathbf{h}_{0:T})$$

$$(\log y)' = \frac{y'}{y}$$

$$= \sum_{\mathbf{a}_{0:T}} \left[\sum_{t=0}^{T} r_t(\mathbf{a}_t) \right] \nabla \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_{0:T} | \mathbf{h}_{0:T}) \, \pi_{\boldsymbol{\theta}}(\mathbf{a}_{0:T} | \mathbf{h}_{0:T})$$

$$= \sum_{\mathbf{a}_{0:T}} \left[\sum_{t=0}^{T} r_t(\mathbf{a}_t) \right] \left[\sum_{t=0}^{T} \nabla \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t | \mathbf{h}_{0:t}) \right] \pi_{\boldsymbol{\theta}}(\mathbf{a}_{0:T} | \mathbf{h}_{0:T})$$

$$= \sum_{\mathbf{a}_{0:T}} \left[\sum_{t=0}^{T} \nabla \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t | \mathbf{h}_{0:t}) \sum_{n=t}^{T} r_n(\mathbf{a}_n) \right] \pi_{\boldsymbol{\theta}}(\mathbf{a}_{0:T} | \mathbf{h}_{0:T})$$

Policy gradients using backprop

$$\nabla_{\boldsymbol{\theta}} J^{\pi}(\boldsymbol{\theta}) = \sum_{\mathbf{a}_{0:T}} \left[\sum_{t=0}^{T} \nabla \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t | \mathbf{h}_{0:t}) \sum_{n=t}^{T} r_n(\mathbf{a}_n) \right] \pi_{\boldsymbol{\theta}}(\mathbf{a}_{0:T} | \mathbf{h}_{0:T})$$

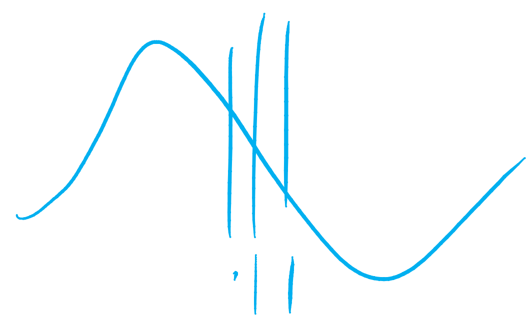
$$\mathbf{a}_{0:T}^{(i)} \sim \pi_{\boldsymbol{\theta}}(\mathbf{a}_{0:T} | \mathbf{h}_{0:T})$$

$$\widehat{\nabla_{\boldsymbol{\theta}} J^{\pi}(\boldsymbol{\theta})} = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t^{(i)} | \mathbf{h}_{0:t}) \sum_{n=t}^{T} r_n(\mathbf{a}_n^{(i)})$$

backprop

samples

Matt Hoffman



Neuro-dynamic programming

Dynamic programming

Csaba Szepesvari

$$\mathbf{a}_t = \pi(\mathbf{s}_t)$$

s denotes the state (model abstraction of the environment, e.g. histories)

Value function

$$V^{\pi}(\mathbf{s}_0) = \mathbb{E}_{\mathbf{s}_1,\mathbf{s}_2,\mathbf{s}_3,\ldots}\left[\sum_{t=0}^{\infty} \gamma^t r_t(\mathbf{s}_t, \pi(\mathbf{s}_t), \mathbf{s}_{t+1}) \,\middle|\, \mathbf{s}_0\right]$$

a_t

V depends on π

discount factor $\gamma \in (0,1)$

optimal
best value function

$$V^{\star}(\mathbf{s}_0) = \max_{\pi} \mathbb{E}_{\mathbf{s}_1,\mathbf{s}_2,\mathbf{s}_3,\ldots}\left[\sum_{t=0}^{\infty} \gamma^t r_t(\mathbf{s}_t, \pi(\mathbf{s}_t), \mathbf{s}_{t+1}) \,\middle|\, \mathbf{s}_0\right]$$

Dynamic programming

$$V^\star(\mathbf{s}_0) = \max_{\pi} \mathbb{E}_{\mathbf{s}_1,\mathbf{s}_2,\mathbf{s}_3,\ldots} \left[\sum_{t=0}^{\infty} \gamma^t r_t(\mathbf{s}_t, \pi(\mathbf{s}_t), \mathbf{s}_{t+1}) \middle| \mathbf{s}_0 \right]$$

$$= \max_{\pi} \mathbb{E}_{\mathbf{s}_1,\mathbf{s}_2,\mathbf{s}_3,\ldots} \left[r_0(\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1) + \sum_{t=1}^{\infty} \gamma^t r_t(\mathbf{s}_t, \pi(\mathbf{s}_t), \mathbf{s}_{t+1}) \middle| \mathbf{s}_0 \right]$$

$$= \max_{\mathbf{a}_0,\pi} \mathbb{E}_{\mathbf{s}_1,\mathbf{s}_2,\mathbf{s}_3,\ldots} \left[r_0(\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1) + \sum_{t=1}^{\infty} \gamma^t r_t(\mathbf{s}_t, \pi(\mathbf{s}_t), \mathbf{s}_{t+1}) \middle| \mathbf{s}_0 \right]$$

$$= \max_{\mathbf{a}_0} \mathbb{E}_{\mathbf{s}_1} \left[r_0(\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1) + \max_{\pi} \mathbb{E}_{\mathbf{s}_2,\mathbf{s}_3,\mathbf{s}_4,\ldots} \left\{ \sum_{t=1}^{\infty} \gamma^t r_t(\mathbf{s}_t, \pi(\mathbf{s}_t), \mathbf{s}_{t+1}) \middle| \mathbf{s}_1 \right\} \middle| \mathbf{s}_0 \right]$$

$$= \max_{\mathbf{a}_0} \mathbb{E}_{\mathbf{s}_1} \left[r_0(\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1) + \gamma \max_{\pi} \mathbb{E}_{\mathbf{s}_2,\mathbf{s}_3,\mathbf{s}_4,\ldots} \left\{ \sum_{t=1}^{\infty} \gamma^{t-1} r_t(\mathbf{s}_t, \pi(\mathbf{s}_t), \mathbf{s}_{t+1}) \middle| \mathbf{s}_1 \right\} \middle| \mathbf{s}_0 \right]$$

$$= \max_{\mathbf{a}_0} \mathbb{E}_{\mathbf{s}_1} \left[r_0(\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1) + \gamma V^\star(\mathbf{s}_1) \right]$$

AB + AC

A(B+C)

$a_0 = \pi(s_0)$

Bellman's equation and TD

current state   future state

$$V^{\star}(\mathbf{s}) = \max_{\mathbf{a}} \mathbb{E}_{\mathbf{s}'}\left[r(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma V^{\star}(\mathbf{s}')|\mathbf{s}\right]$$

$$V^{\pi}(s) + \eta V^{\pi}(s) = \mathbb{E}_{s'}\left[r(s,a,s') + \gamma V^{\pi}(s')\right]\eta + V^{\pi}(s)$$

$$V^{\pi}(s) = V^{\pi}(s) + \eta\left\{\mathbb{E}_{s'}\left[r(s,a,s') + \gamma V^{\pi}(s')\right] - V^{\pi}(s)\right\}$$

$$\tilde{s}' \sim P(s'|\ldots)$$

$$V^{\pi}_{t+1}(s) = V^{\pi}_{t}(s) + \eta\left\{r(s,a,\tilde{s}') + \gamma V^{\pi}_{t}(\tilde{s}')\right\} - V^{\pi}_{t}(s)$$

Action-value (Q) functions

$$V^{\pi}(\mathbf{s}) = \max_{\mathbf{a}'} Q^{\pi}(\mathbf{s}, \mathbf{a}')$$

$$V^{\star}(\mathbf{s}) = \max_{\mathbf{a}} \mathbb{E}_{\mathbf{s}'} \left[r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma V^{\star}(\mathbf{s}') | \mathbf{s} \right]$$

$$V^{\star}(\mathbf{s}) = \max_{\mathbf{a}} \mathbb{E}_{\mathbf{s}'} \left[r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma \max_{\mathbf{a}'} Q^{\star}(\mathbf{s}', \mathbf{a}') | \mathbf{s} \right]$$

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

Q - Learning

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

Neuro-dynamic programming

$Q(\mathbf{s}, \mathbf{a}; \mathbf{w})$, where $\mathbf{w}$ are the parameters

$$L(\mathbf{w}_i) = \mathbb{E}_{\mathbf{s},\mathbf{a}} \left\{ \left(\mathbb{E}_{\mathbf{s}'} \left[r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}', \mathbf{a}', \mathbf{w}_{i-1}) \right] - Q(\mathbf{s}, \mathbf{a}, \mathbf{w}_i) \right)^2 \right\}$$

target

critic

actor

$$y_i = \mathbb{E}_{\mathbf{s}'} \left[r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}', \mathbf{a}', \mathbf{w}_{i-1}) \right]$$

target

Neuro-dynamic programming

$$L(\mathbf{w}_i) = \mathbb{E}_{\mathbf{s},\mathbf{a}} \left\{ \left(\mathbb{E}_{\mathbf{s}'} \left[r(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}',\mathbf{a}',\mathbf{w}_{i-1}) \right] - Q(\mathbf{s},\mathbf{a},\mathbf{w}_i) \right)^2 \right\}$$

$$\nabla_{\mathbf{w}_i} L(\mathbf{w}_i) = \mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'} \left\{ \left(r(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}',\mathbf{a}',\mathbf{w}_{i-1}) - Q(\mathbf{s},\mathbf{a},\mathbf{w}_i) \right) \nabla_{\mathbf{w}_i} Q(\mathbf{s},\mathbf{a},\mathbf{w}_i) \right\}$$

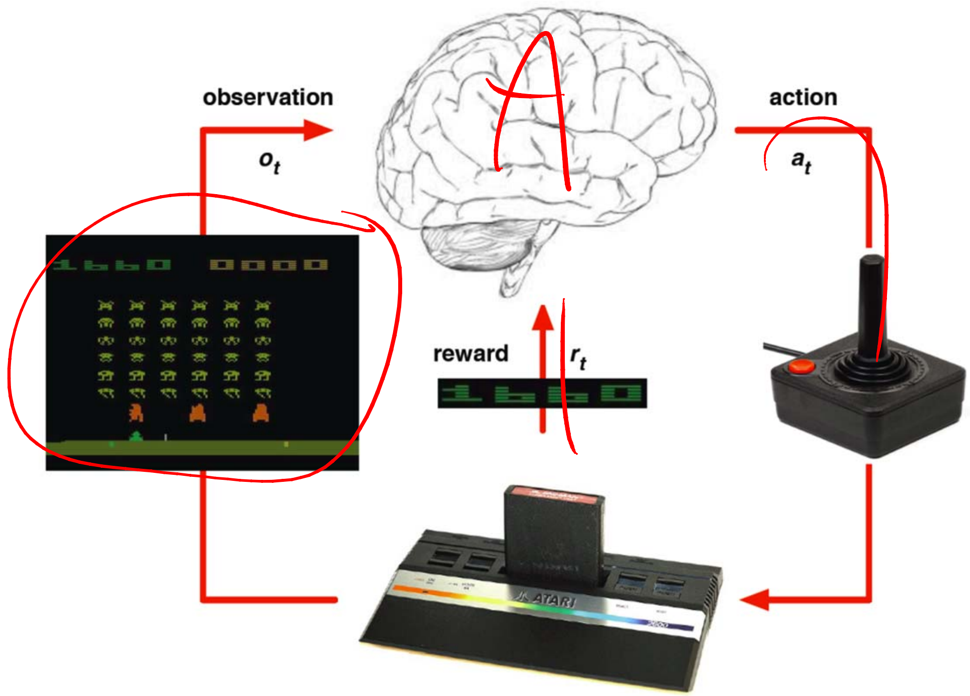
$$\widehat{\mathbf{a}} = \arg\max_{\mathbf{a}} Q(\mathbf{s},\mathbf{a};\widehat{\mathbf{w}})$$

1. state s
2. Choose a
3. Environment → s'
4. Update w

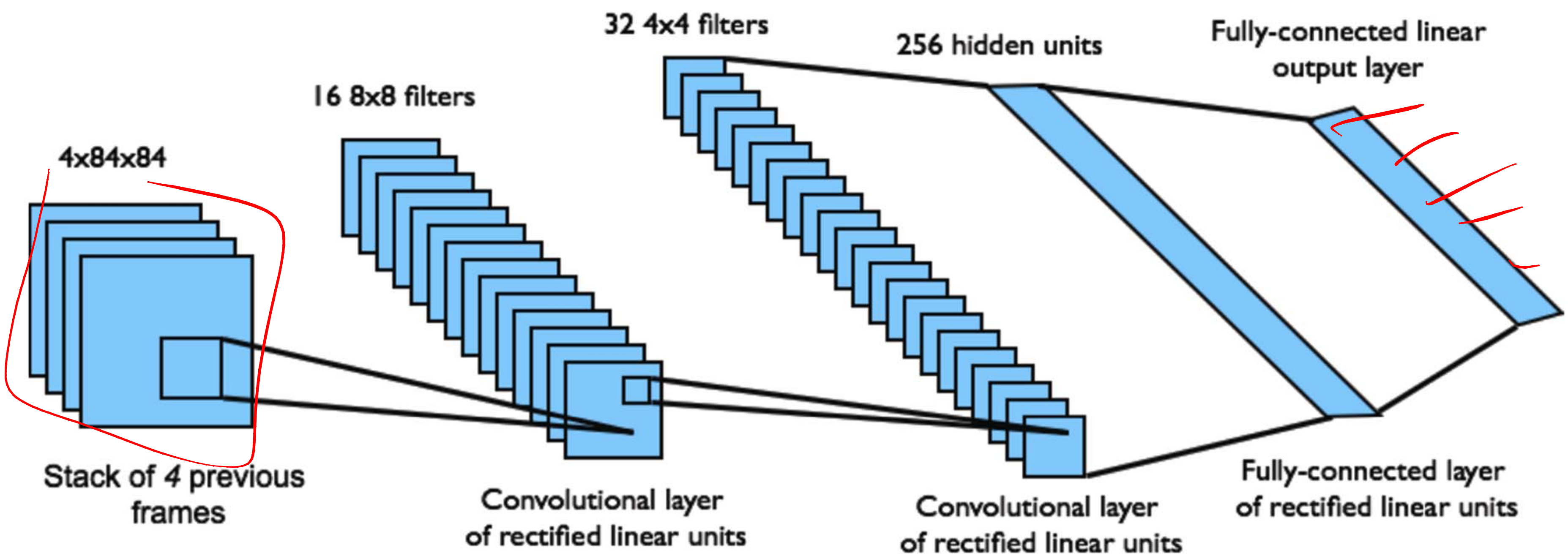
$\begin{cases} \text{w.p. } \varepsilon & a = \max_{a'} Q(s,a') \\ \text{w.p. } 1-\varepsilon & a \sim U[\;] \end{cases}$

Deepmind's DQN

- Use a deep neural network to represent the action-value function *Q*.
- Learn *Q* with end-to-end RL mapping the raw pixels to action values in Atari games.
- New stable online variant of Q-learning:
  - Do updates on samples of past experience.
  - Freeze network used for generating targets and refresh periodically.

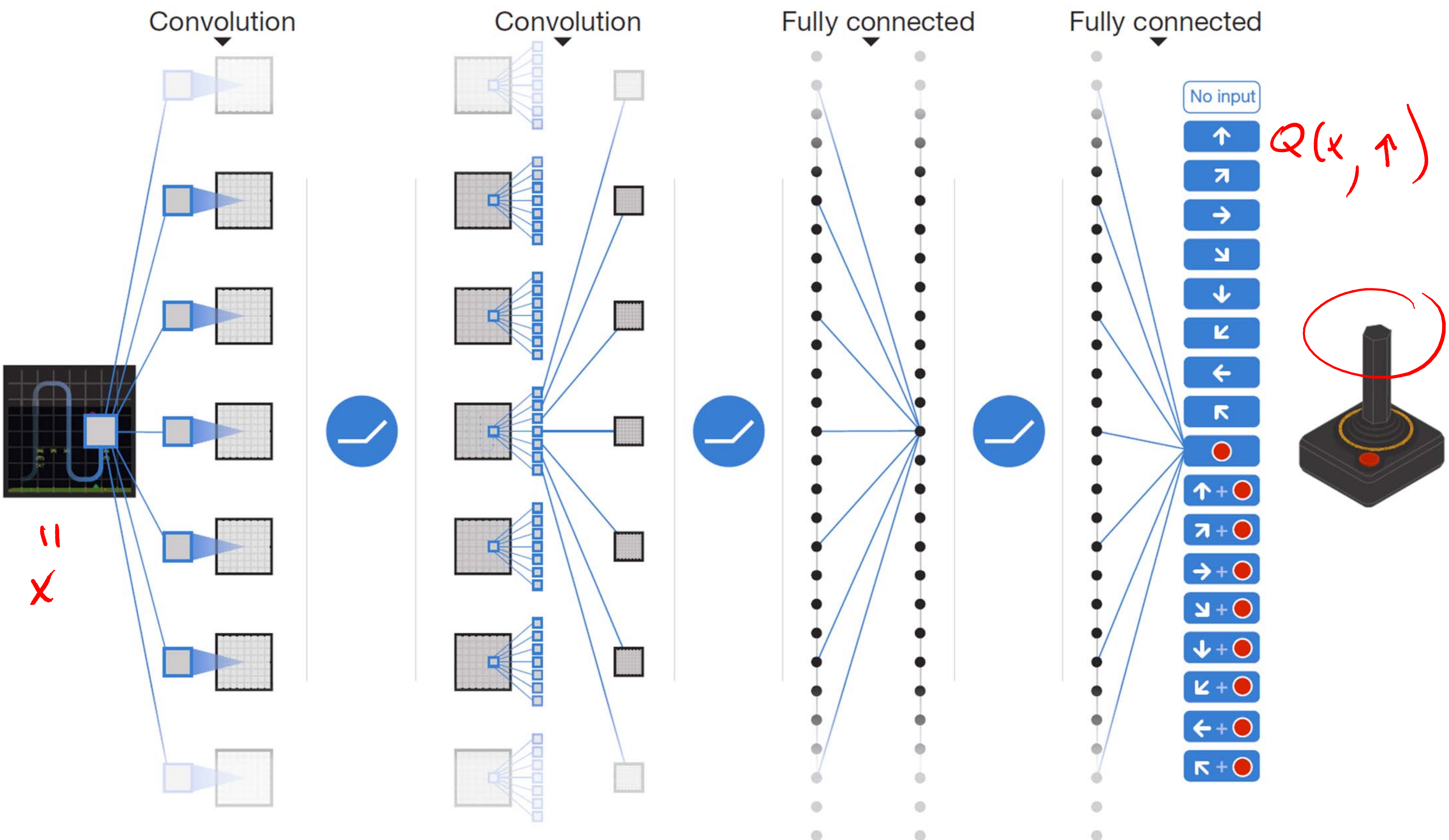


- End-to-end learning of values $Q(s, a)$ from pixels s
- Input state s is stack of raw pixels from last 4 frames
- Output is $Q(s, a)$ for 18 joystick/button positions
- Reward is change in score for that step



Network architecture and hyperparameters fixed across all games

DQN Convolutional Network



Q-learning with experience re-play

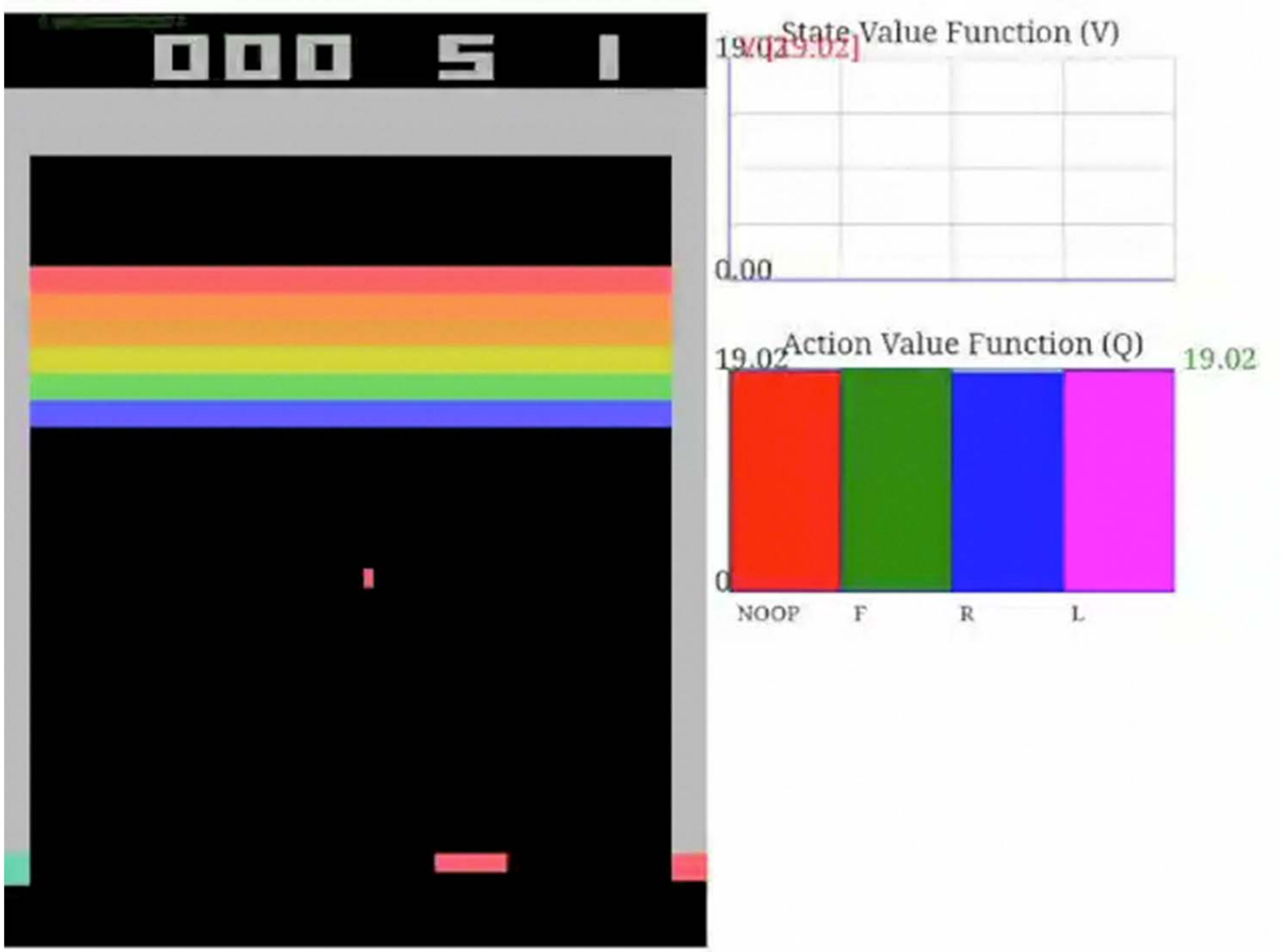
DQN increases stability with experience replay and fixed Q-targets

- Take action a_t according to ϵ-greedy policy
- Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in replay memory $\mathcal{D}$
- Sample random mini-batch of transitions (s, a, r, s') from $\mathcal{D}$
- Compute Q-learning targets w.r.t. old, fixed parameters w^-
- Optimise MSE between Q-network and Q-learning targets

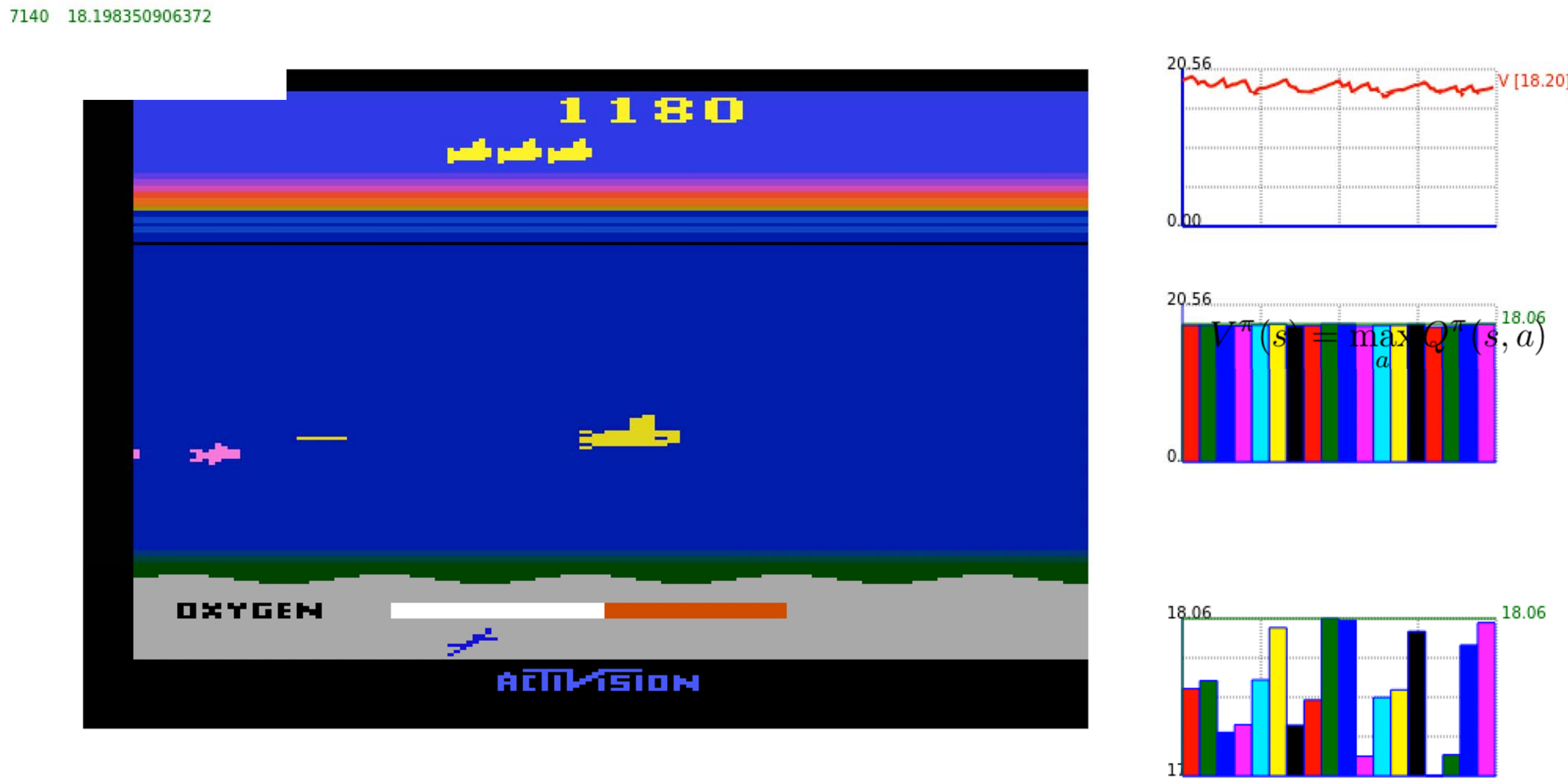
$$\mathcal{L}_i(w_i) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}_i}\left[\left(r + \gamma \max_{a'} Q(s', a'; w_i^-) - Q(s, a; w_i)\right)^2\right]$$

- Using variant of stochastic gradient descent (Mnih et al.)

Delayed Rewards

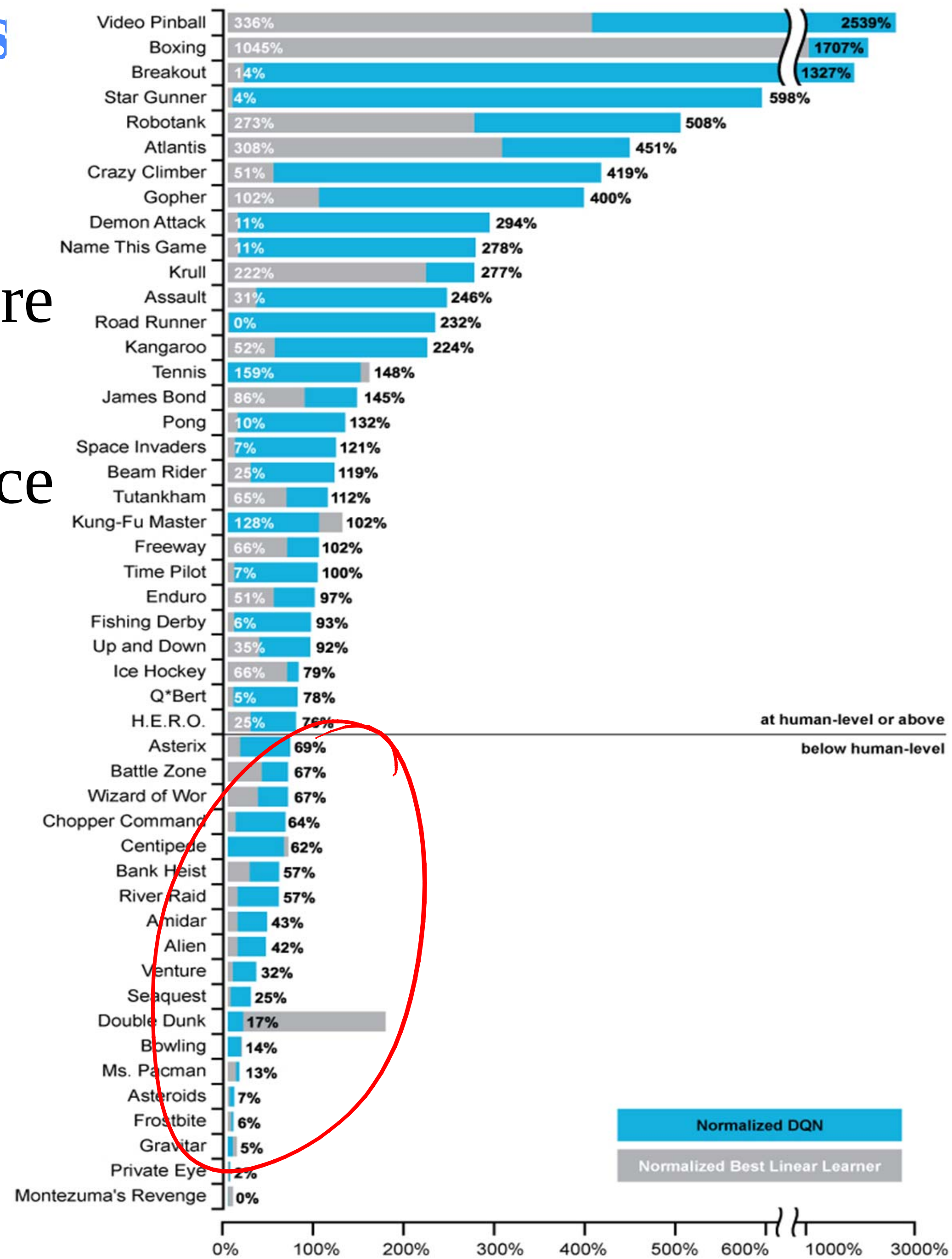


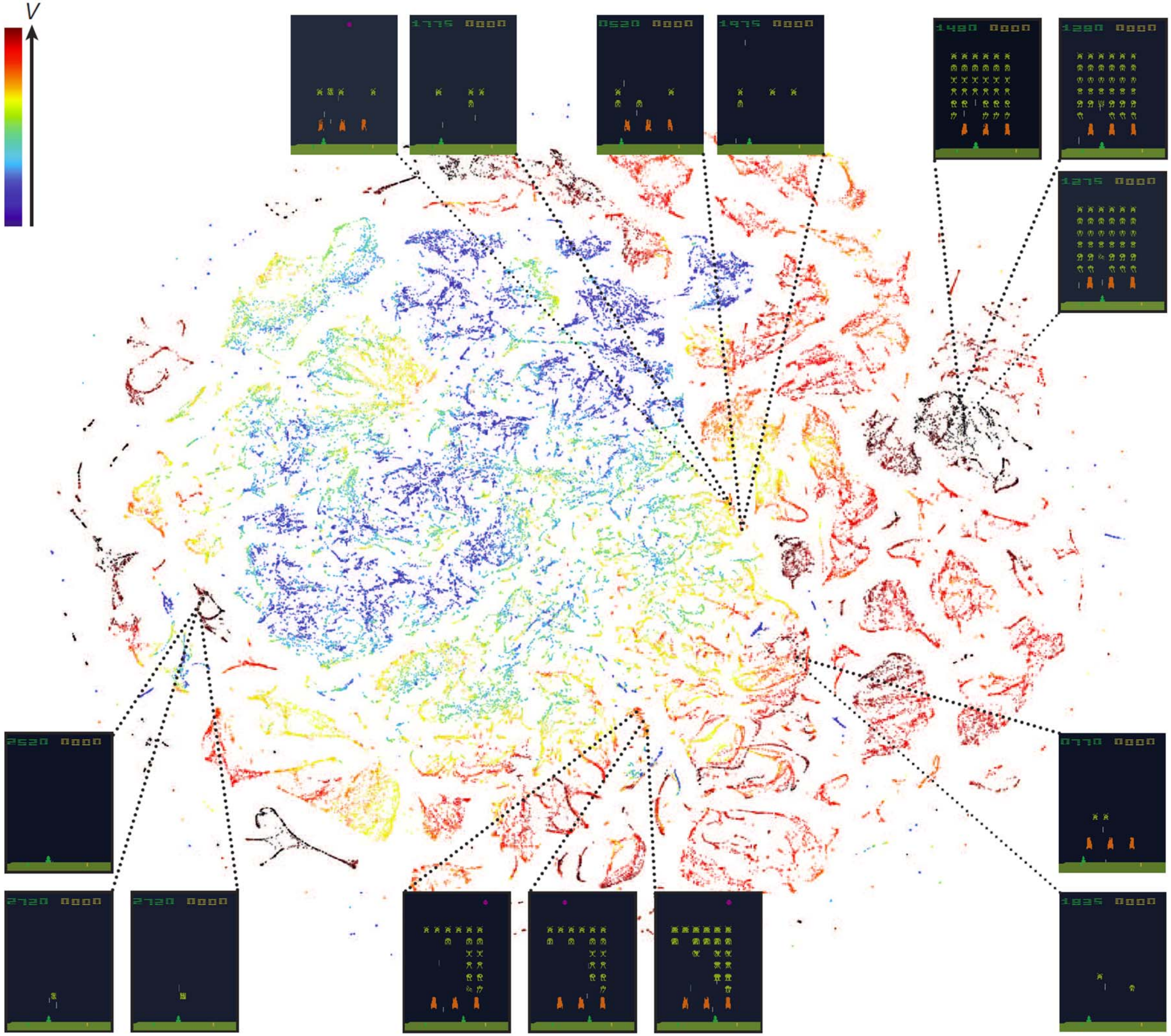
Sacrificing Immediate Rewards



Results on 49 Games

- The architecture and hyperparameter values were the same for all 49 games.
- DQN achieved performance **comparable to or better than an experienced human** on **29 out of 49** games.
- Games with superhuman level play include Pong, Boxing, Breakout, Space Invaders.




V
1775 0000
0520 0000
1975 0000
1480 0000
1280 0000
1275 0000
2520 0000
2720 0000
2720 0000
0770 0000
1835 0000

DQN
AI
And all that

Thank you