Reinforcement Learning

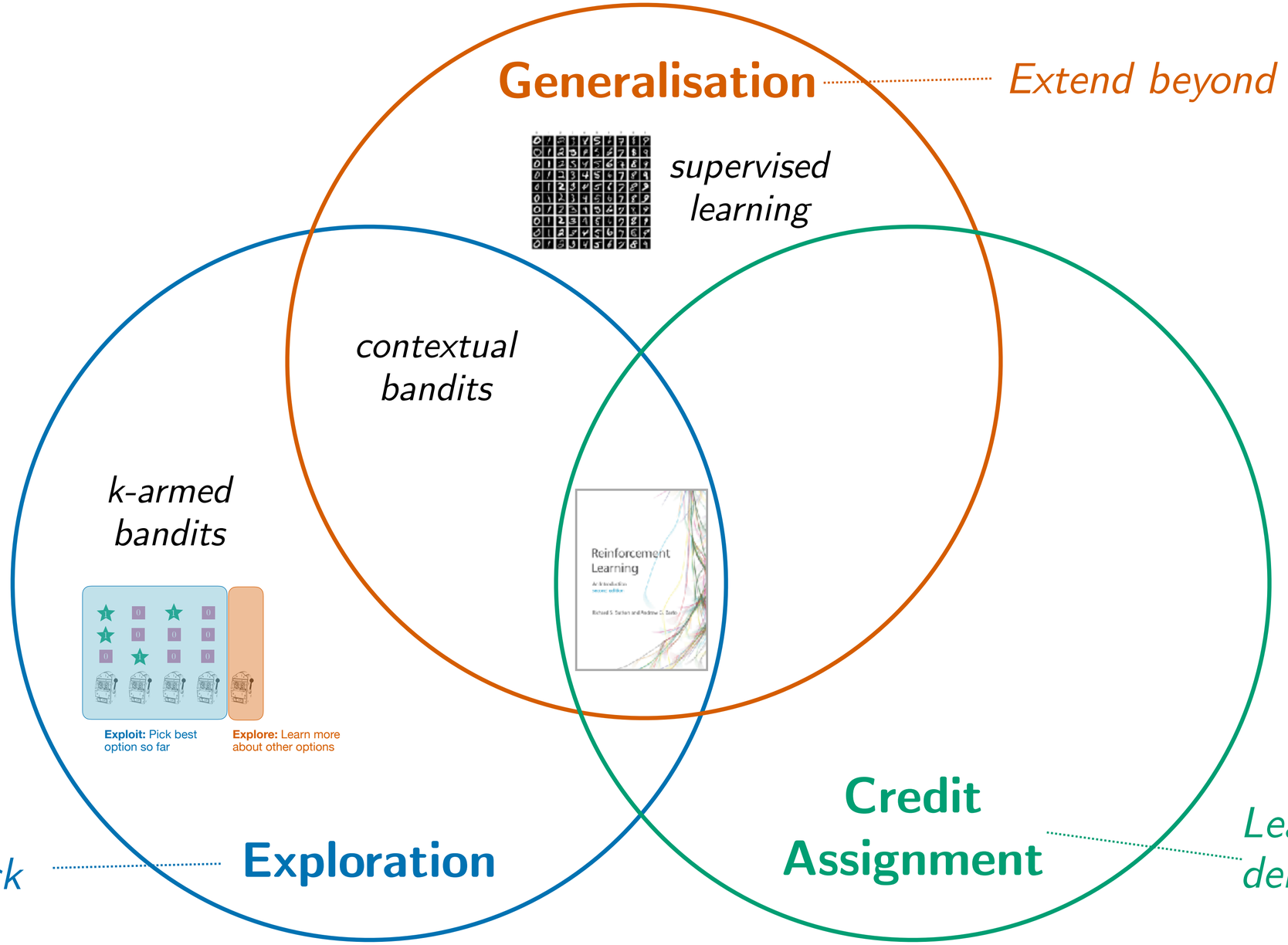
Deep Reinforcement Learning

David Abel, Michael Herrmann
Based on slides by Stefano V. Albrecht
4 March, 2025

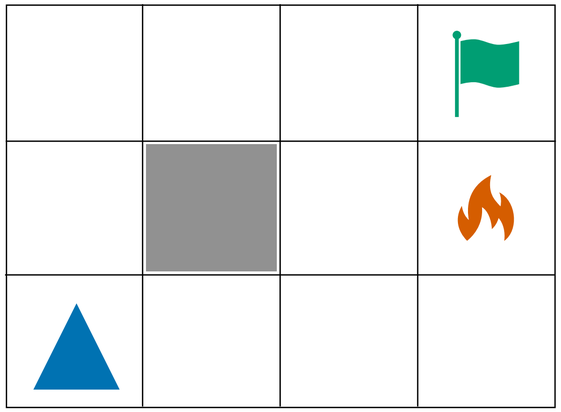
Lecture Outline

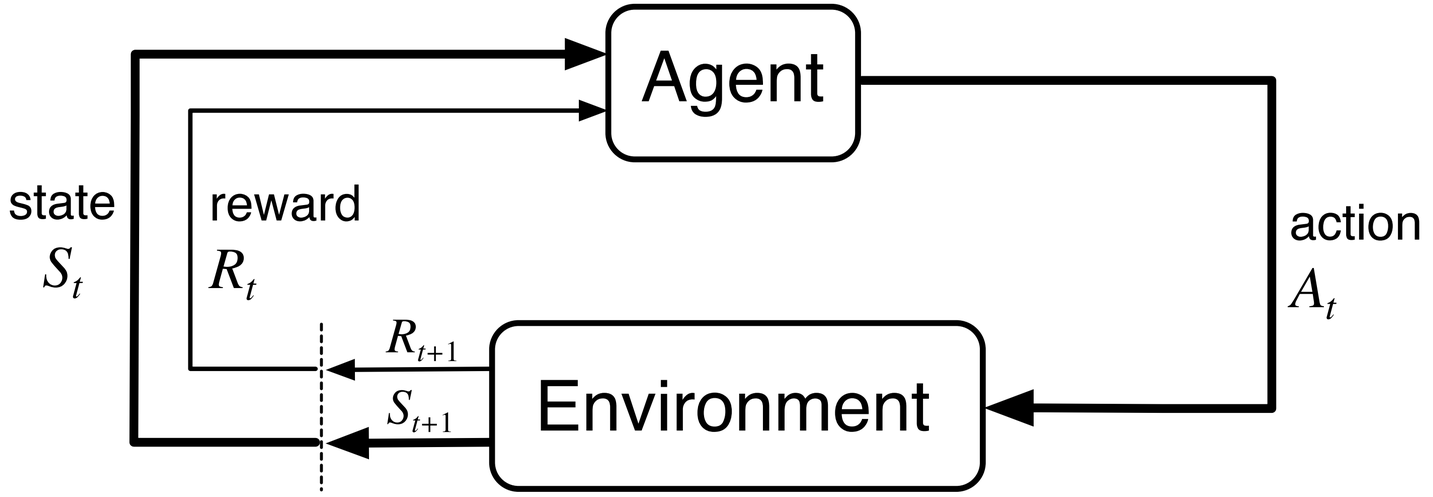
1. Neural Nets
2. Deep RL = RL + Neural Nets
3. Algorithms: DQN, TRPO/PPO

The Three Challenges of RL

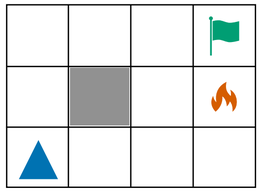
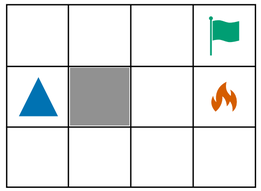
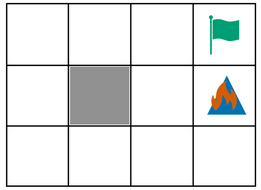


The Reinforcement Learning Loop

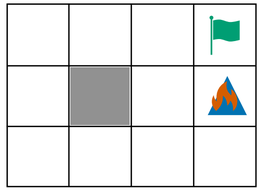




“Tabular” Reinforcement Learning

s	$V(s)$
	0.7
	0.6
	-0.1

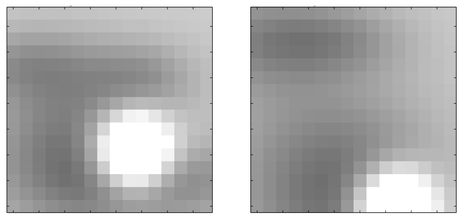
$$Q_{t+1}(s,a) = (1-\alpha)Q_t(s,a) + \alpha(r + \gamma \max_{a'} Q_t(s',a'))$$

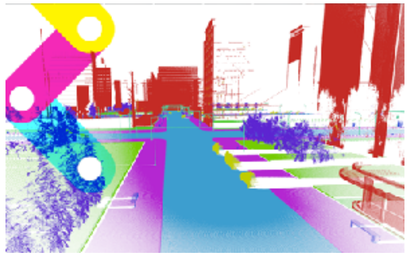


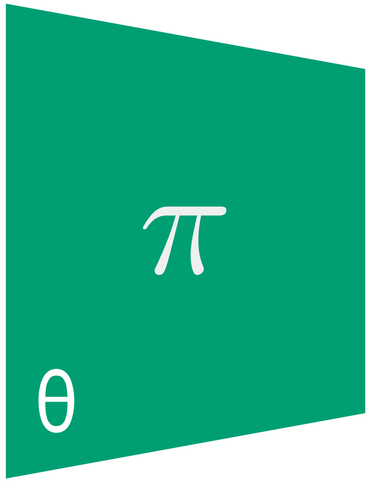
-1.0

New Experience

The Promise of Deep Reinforcement Learning

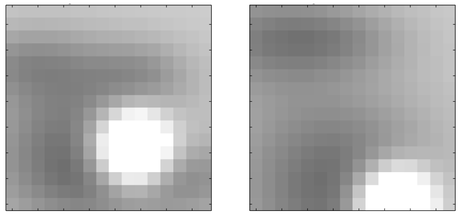


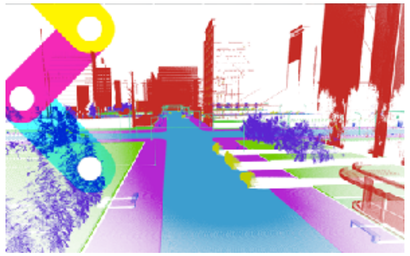


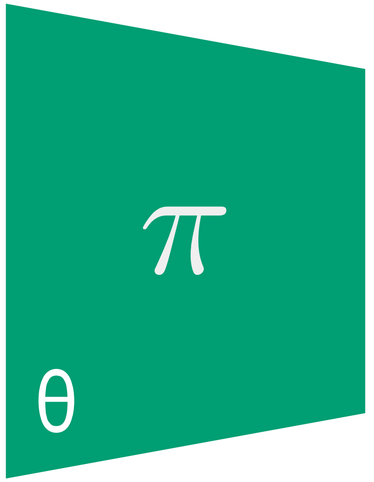


action

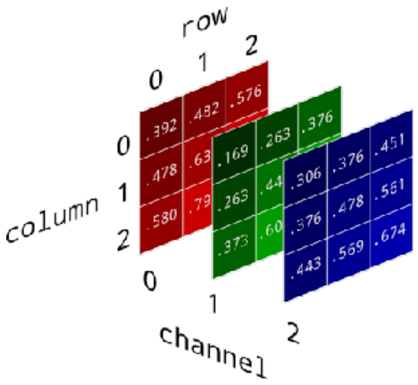
The Promise of Deep Reinforcement Learning







action



=

Recap: Linear Value Function Approximation

$$\hat{v}(s, \mathrm{w}) = \mathrm{w}^\top x(s) = \sum_{i=1}^{d} \mathrm{w}_i x_i(s)$$

Linear representation of value

$$\mathrm{w}_{t+1} = \mathrm{w}_t + \alpha[U_t - \hat{v}(S_t, \mathrm{w}_t)]x(S_t)$$

gradient-based update

$$x(s) = \langle x_1(s), x_2(s), \ldots, x_d(s) \rangle$$

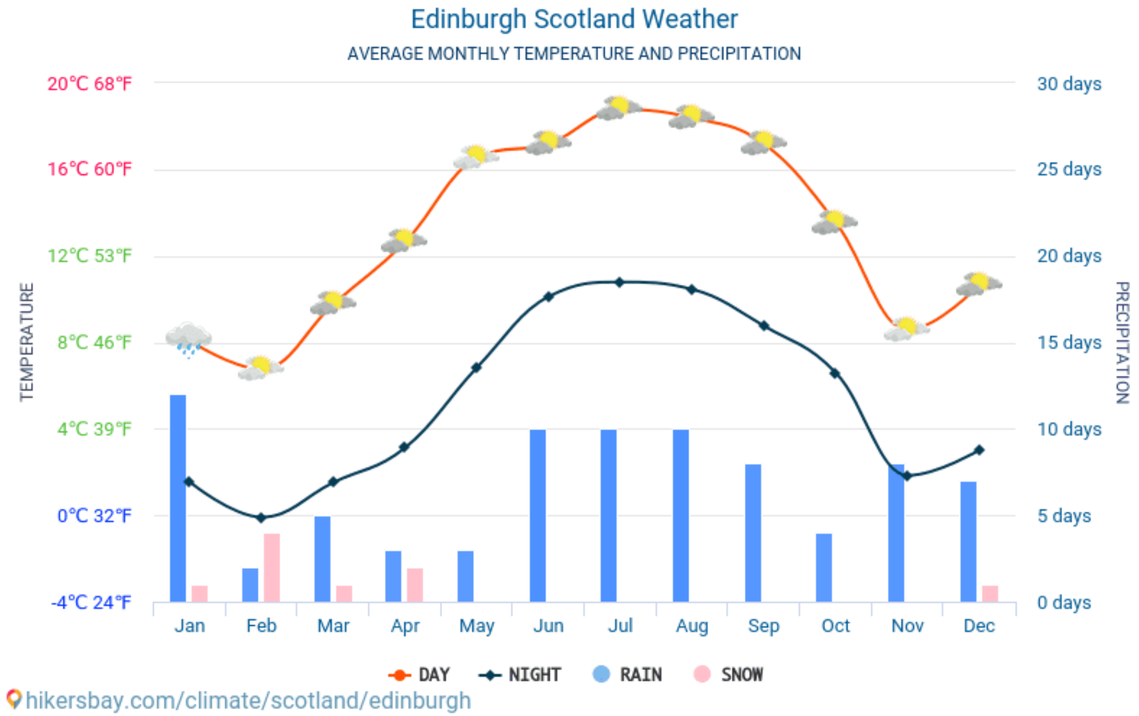
state features

Issue: Linearity!

$$\hat{v}(s, \mathrm{w}) = \mathrm{w}^\top x(s) = \sum_{i=1}^{d} \mathrm{w}_i x_i(s)$$

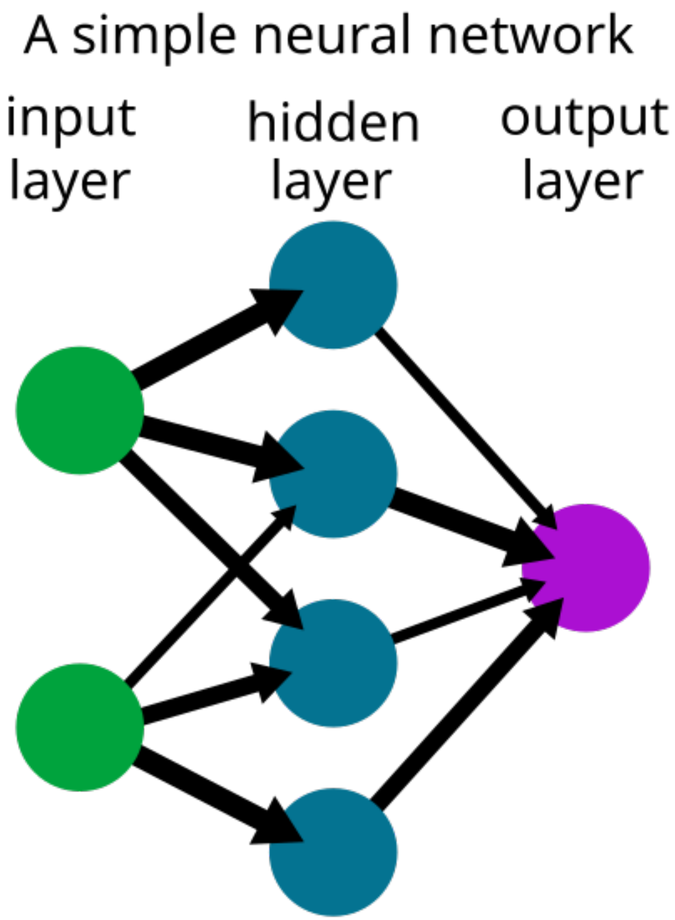
Linear representation of value

Linear is: simple!

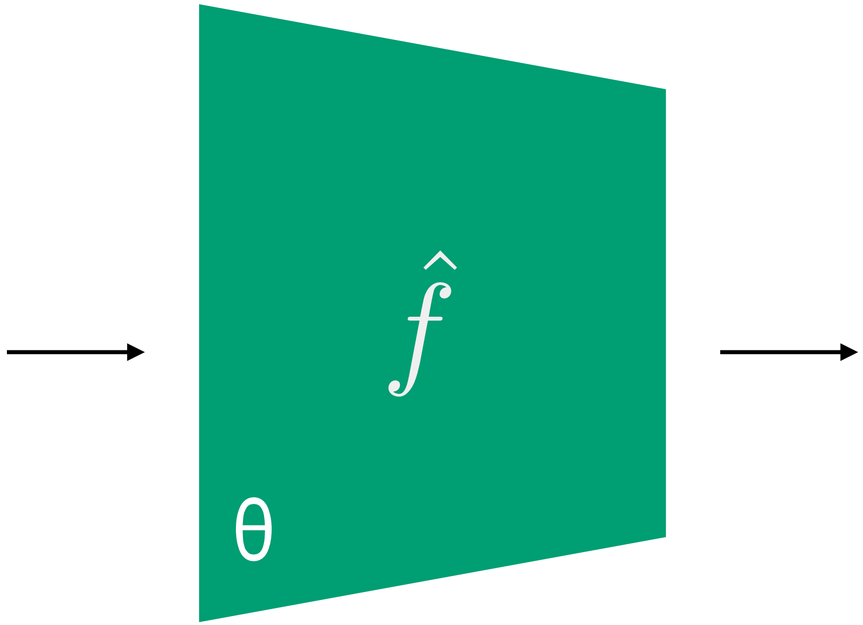


But, limited...

Enter: Neural Nets



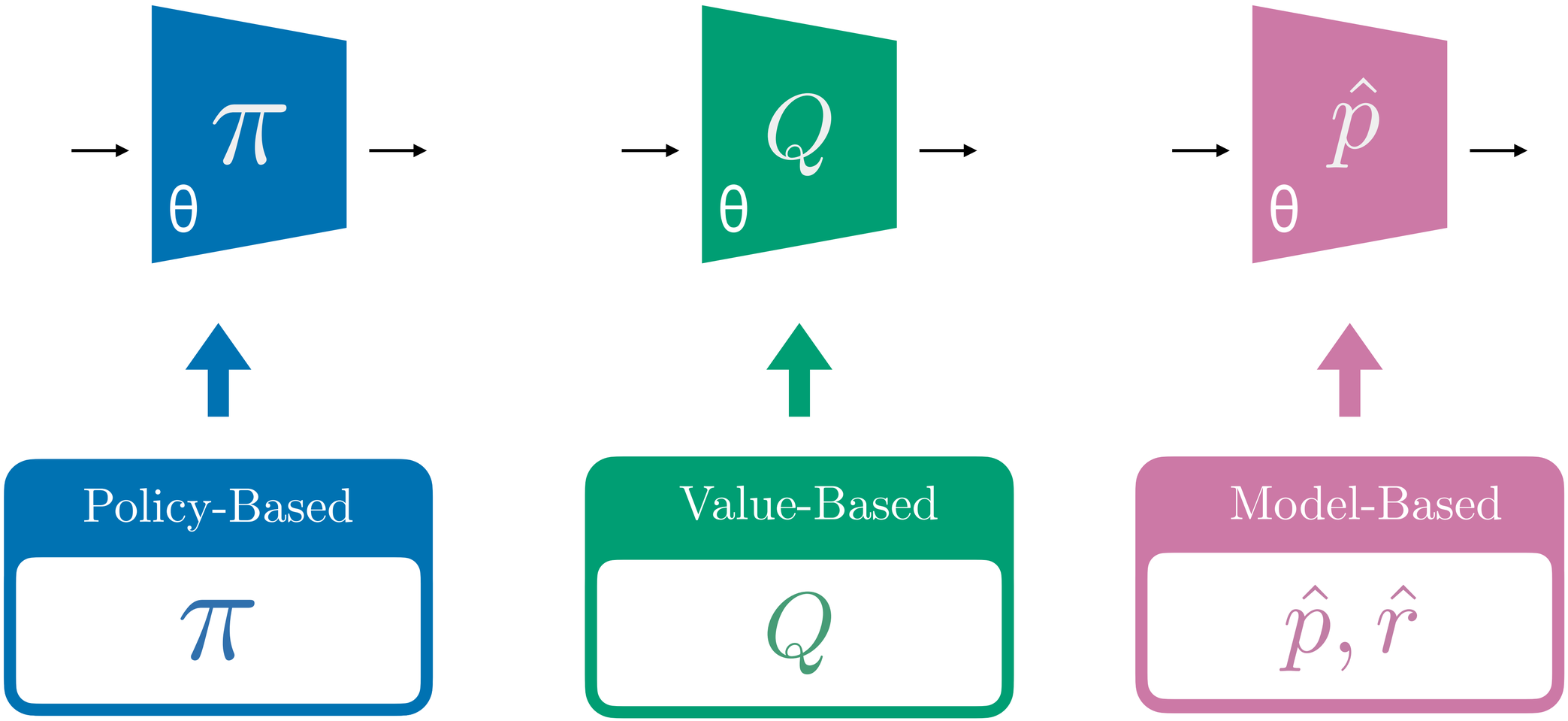
Enter: Neural Nets — General Purpose Function Approximators



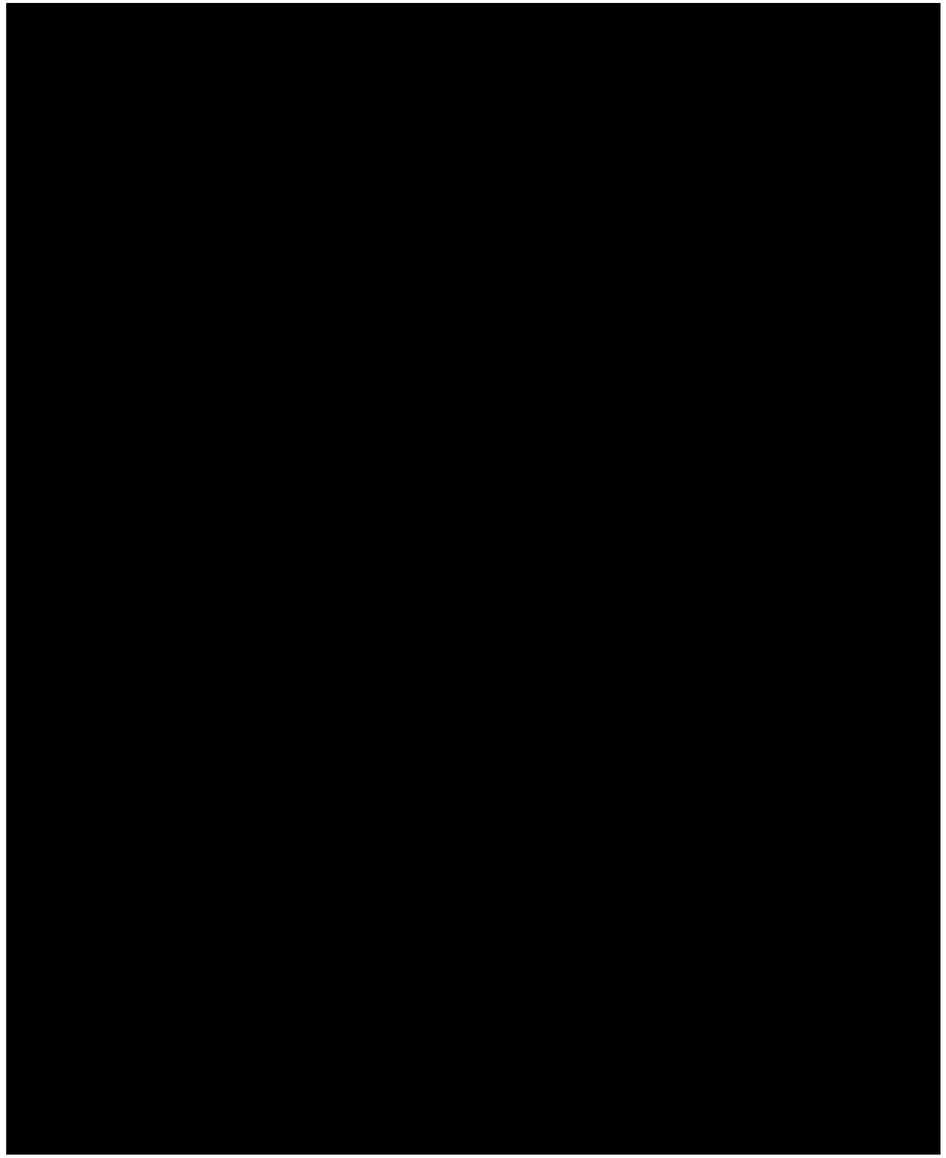
Theorem. Neural Networks are Universal Approximators (Informal).

For any continuous function on the reals $f : \mathbb{R} \to \mathbb{R}$, there will exist a neural network that approximates that function.

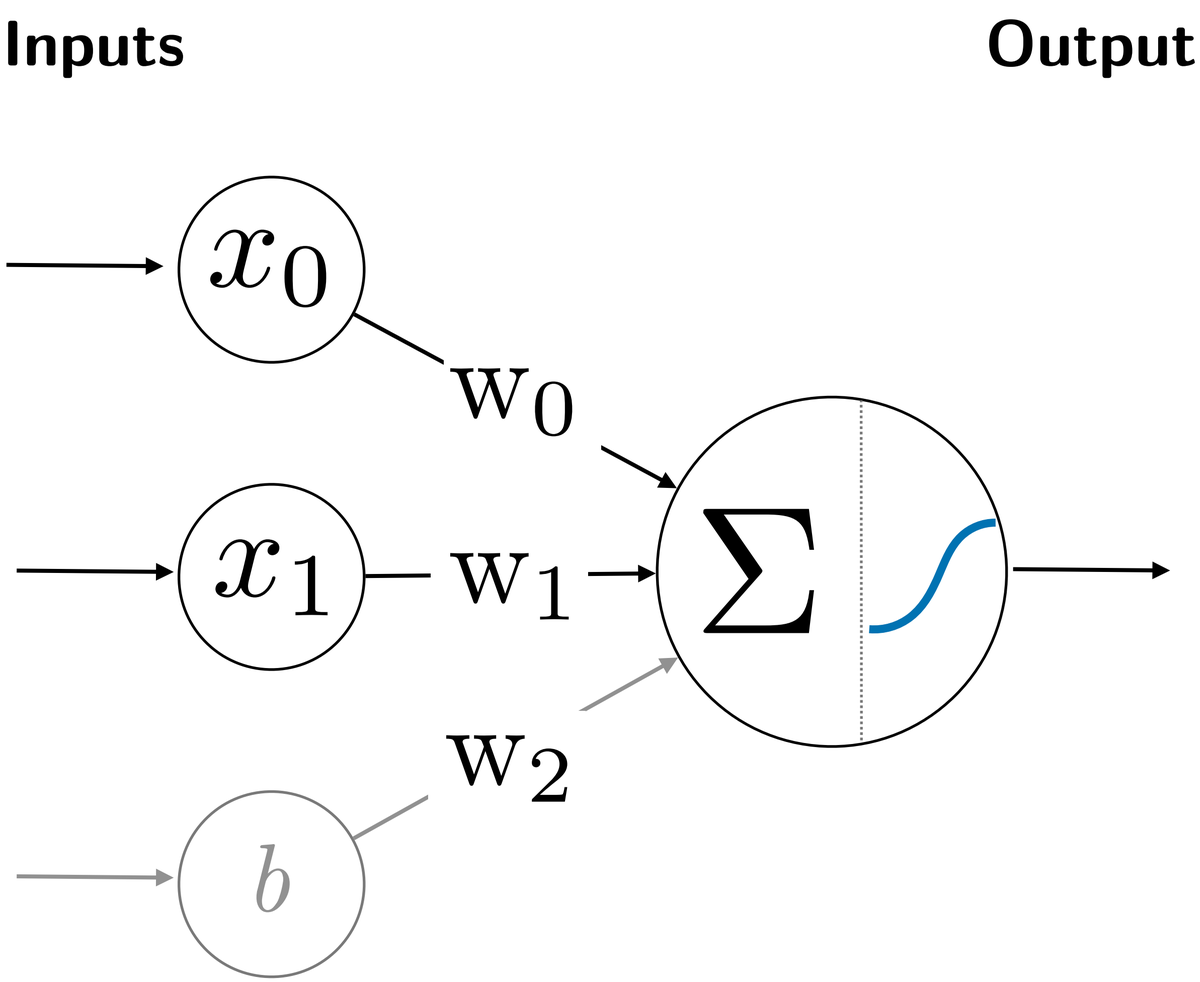
Deep RL


π
θ
Q
θ
$\hat{p}$
θ
Policy-Based
π
Value-Based
Q
Model-Based
$\hat{p}, \hat{r}$

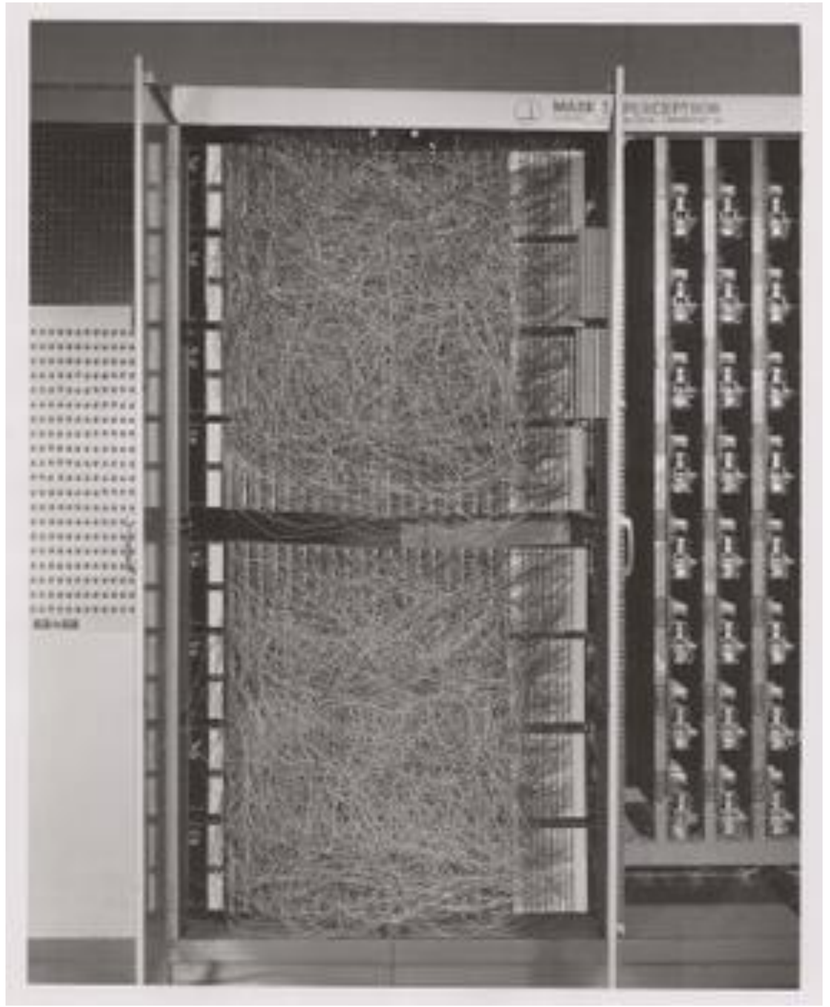
Deep Q-Networks by Mnih et al. (2013, 2015)



Neural Nets: A Brief History

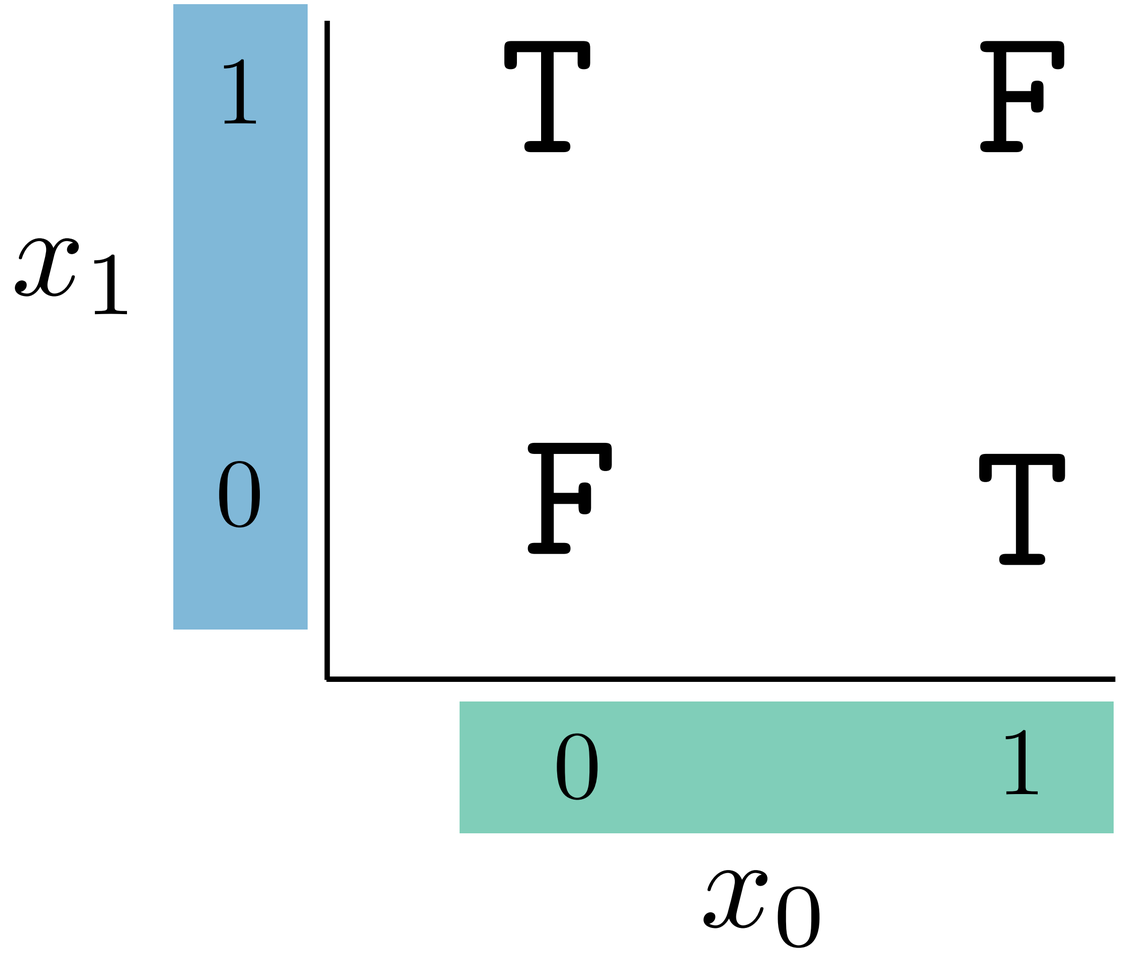


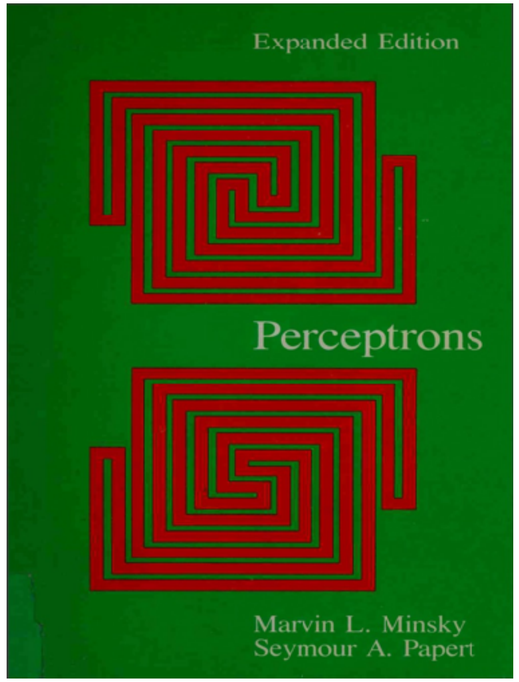
Perceptron: 1943 by McCulloch and Pitts



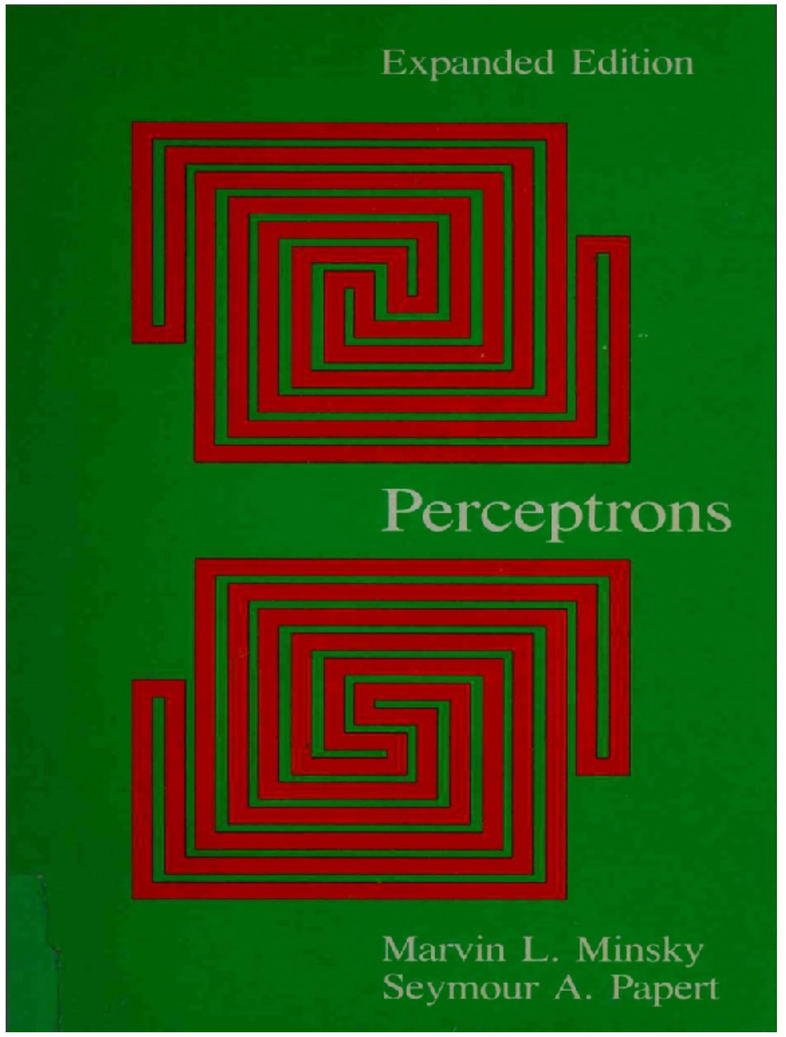
Rosenblatt, 1958

Perceptron and the XOR





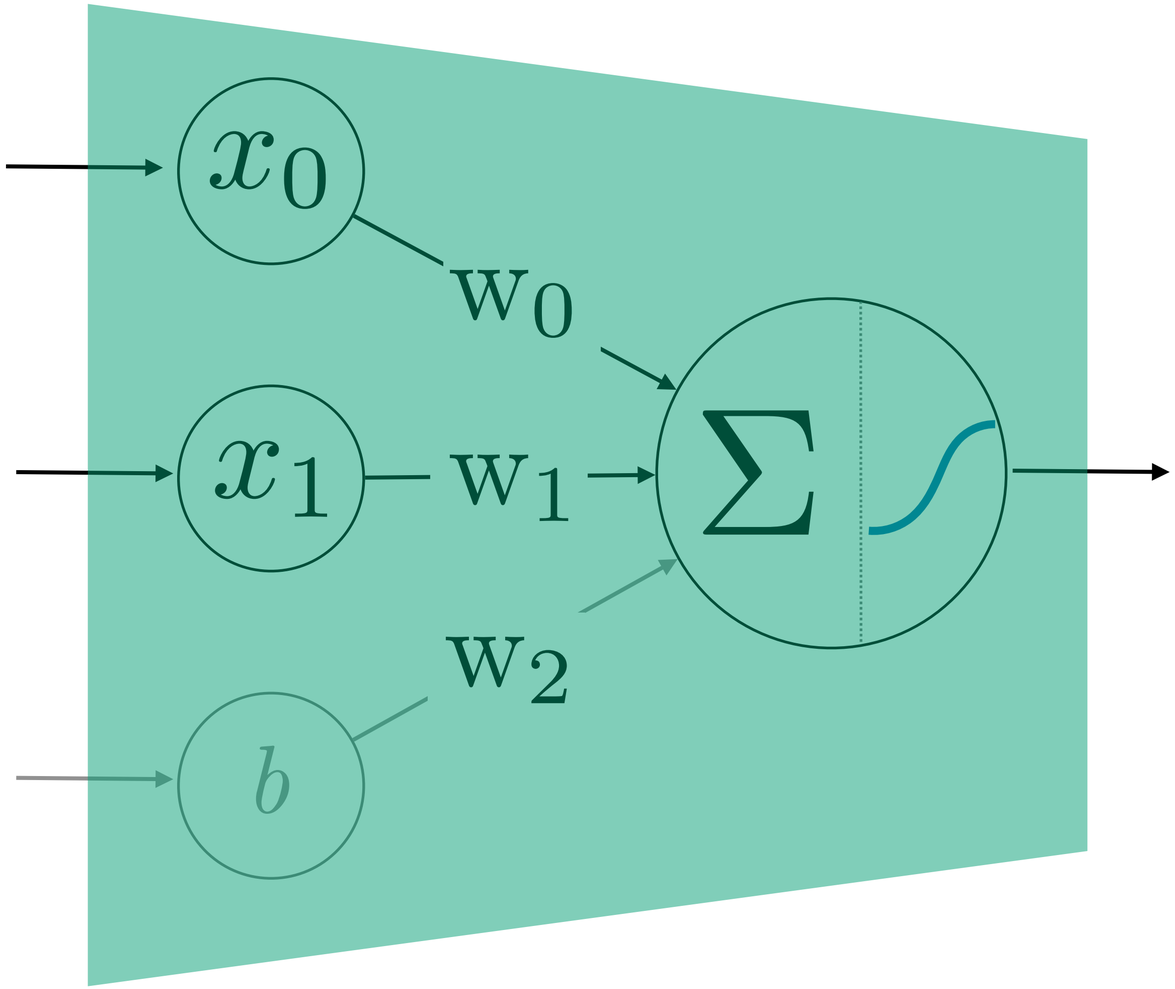
AI Winter: 1974-1980



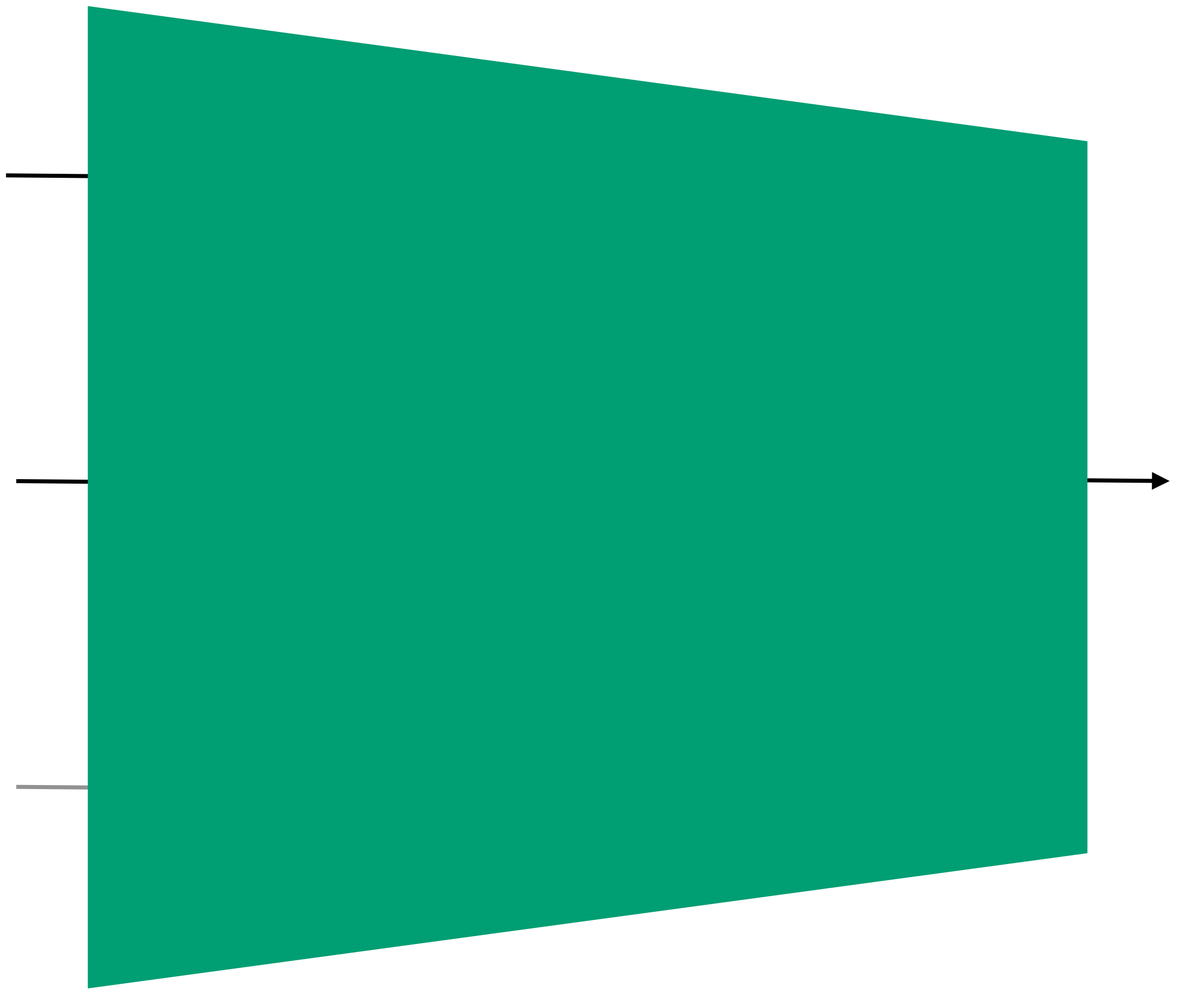
The Lighthill Report

https://www.youtube.com/watch?v=03p2CADwGF8

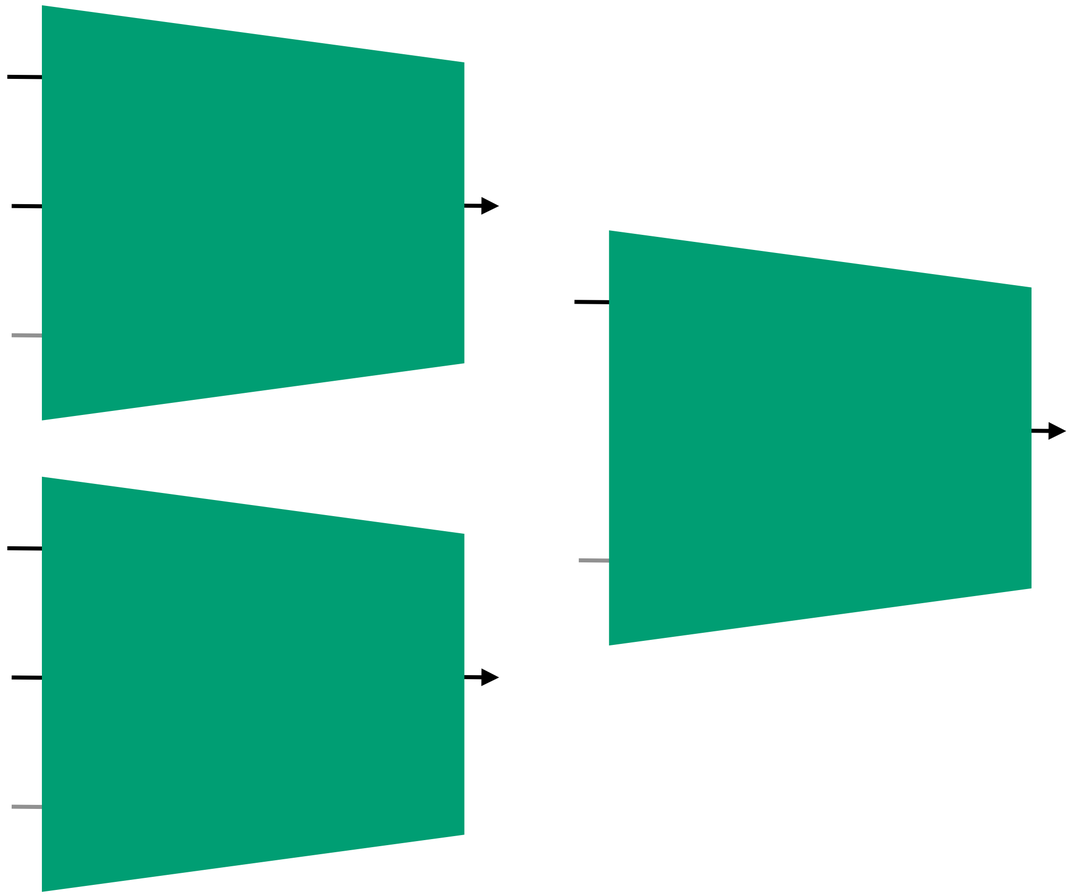
Multilayer Perceptrons



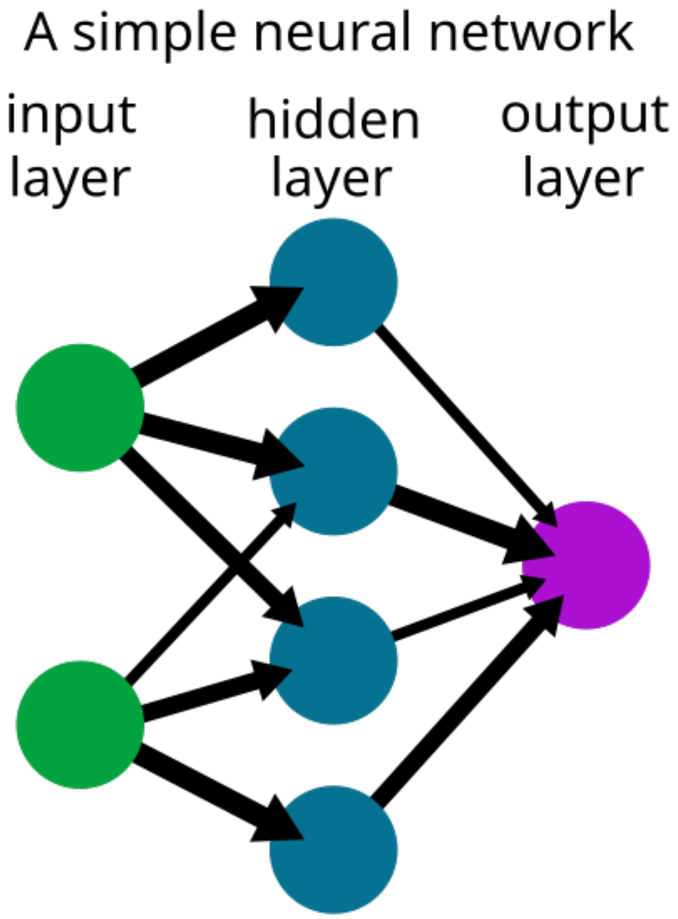
Multilayer Perceptrons



Multilayer Perceptrons

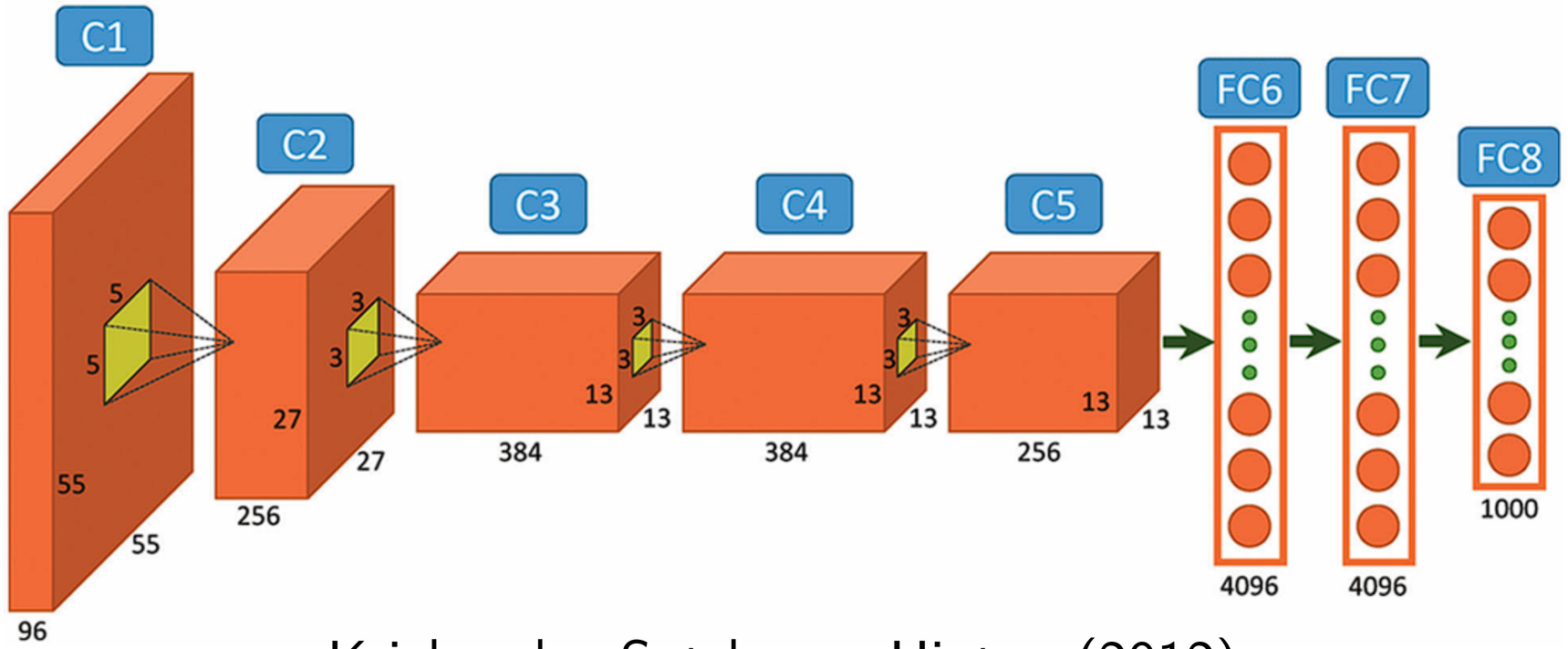


Multilayer Perceptrons



AlexNet, 2010s Boom

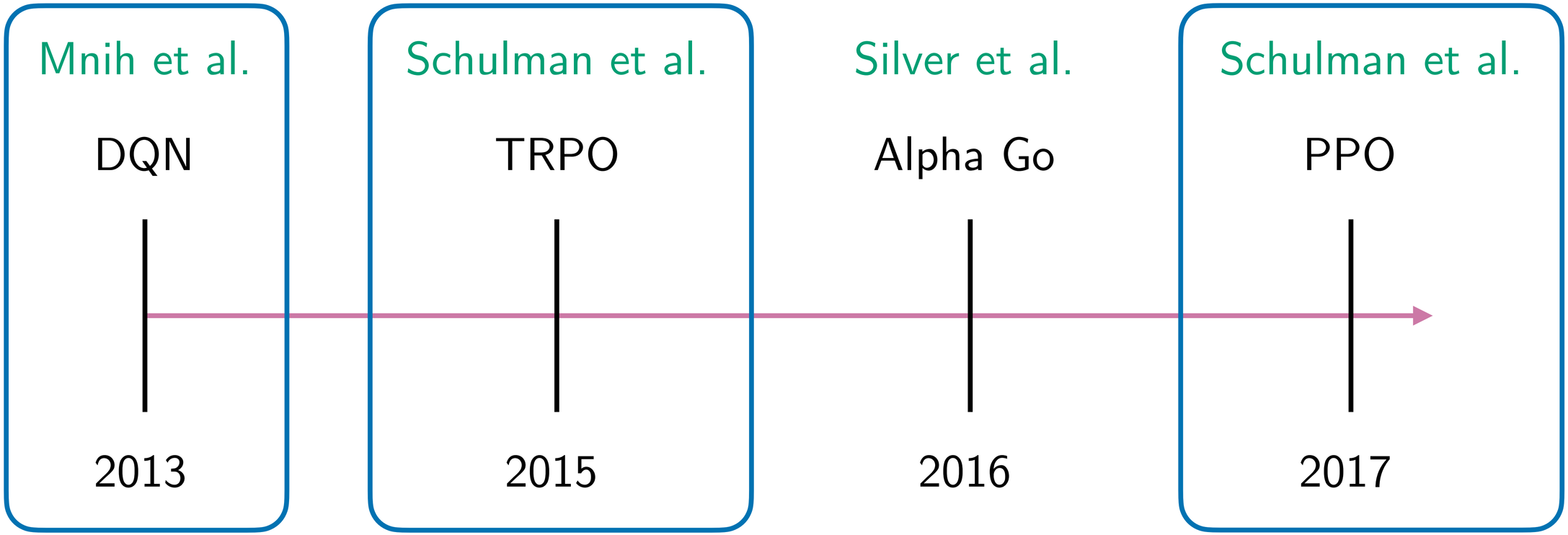
State of the art on computer vision tasks of the time!



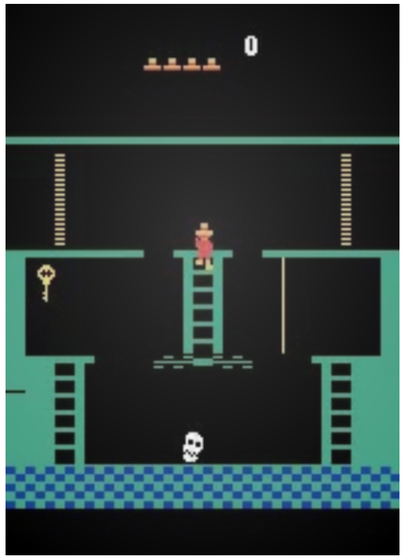
Krizhevsky, Sutskever, Hinton (2012)

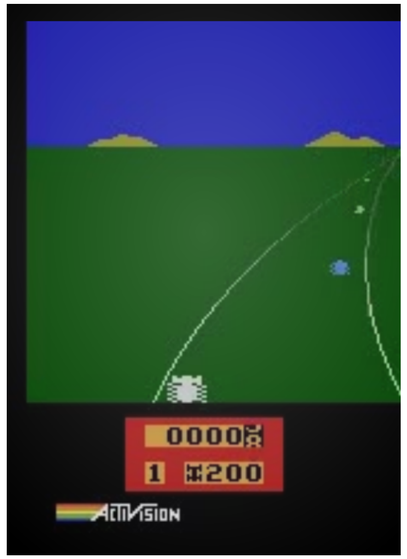
Deep RL: The Algorithmic Journey

+ Double Q-learning, Rainbow, A3C, DDPG, Soft Actor-Critic, Distributional RL...



Deep Q-Networks (DQN, Mnih et al. 2015)



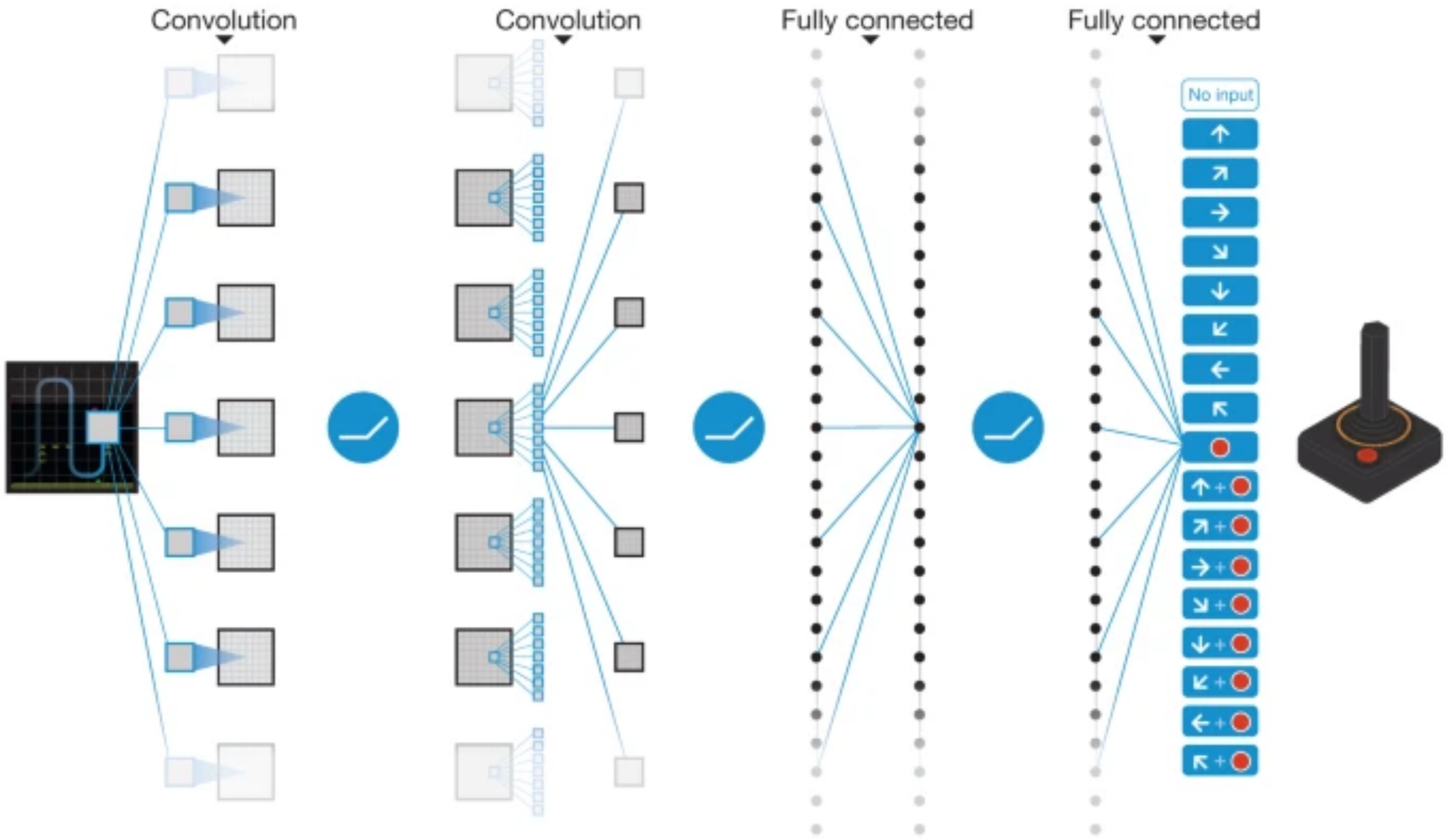


Deep Q-Networks (DQN, Mnih et al. 2015)

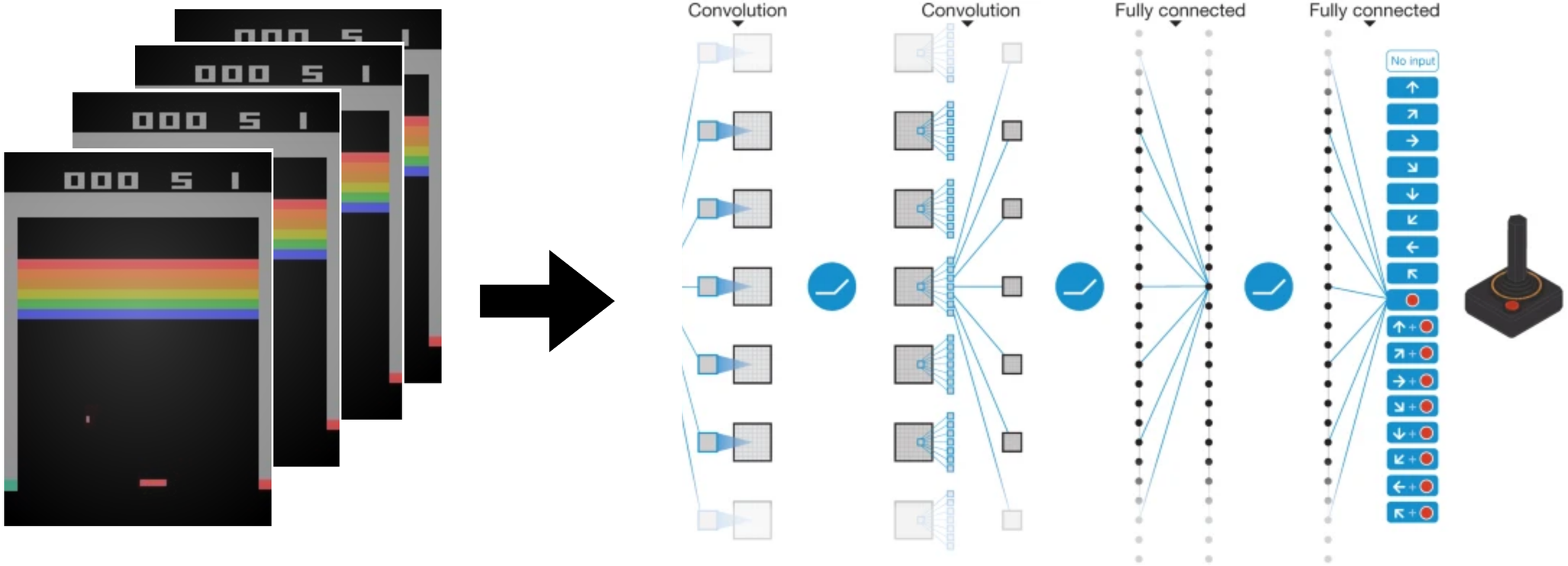
Discussion (2 minutes):

Given an RGB image of breakout as state, does the Markov property hold?

Deep Q-Networks (DQN, Mnih et al. 2015)

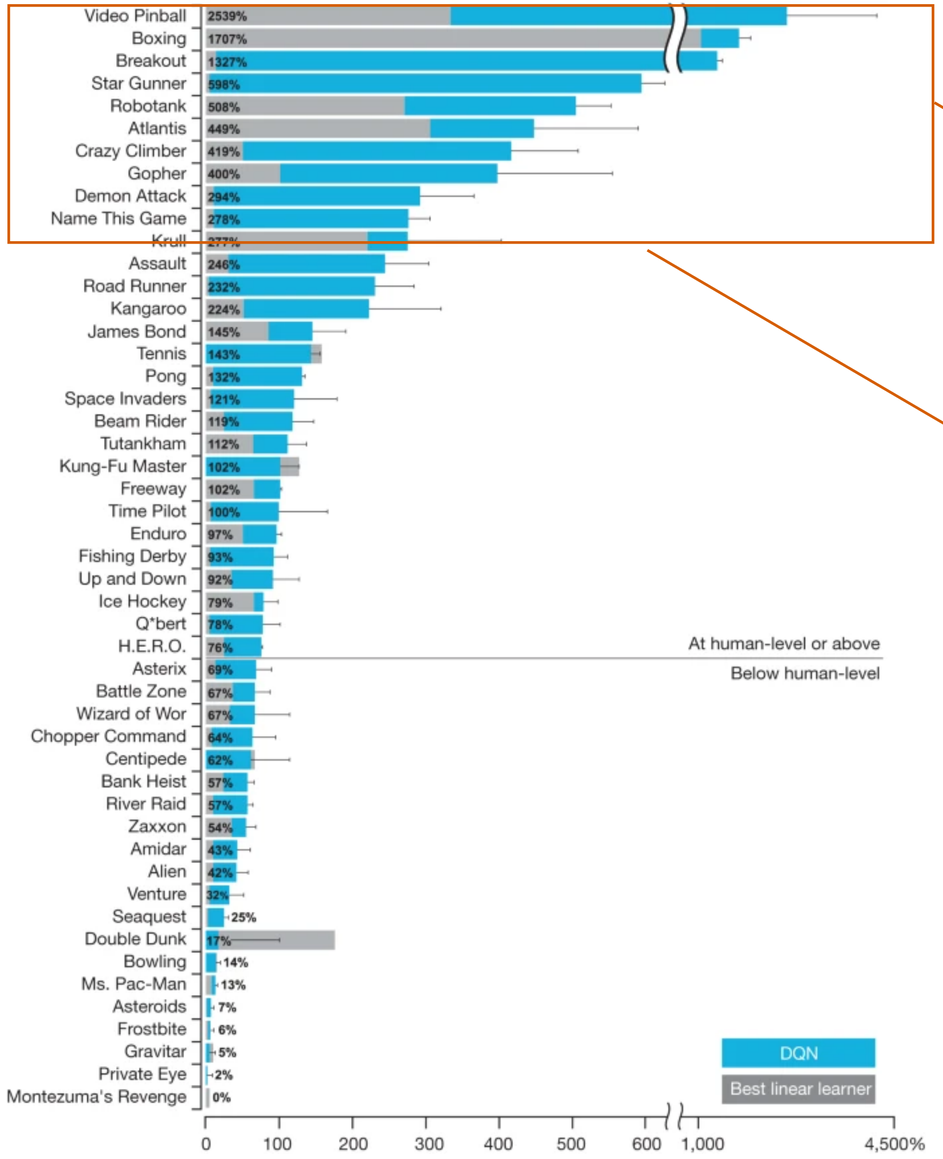


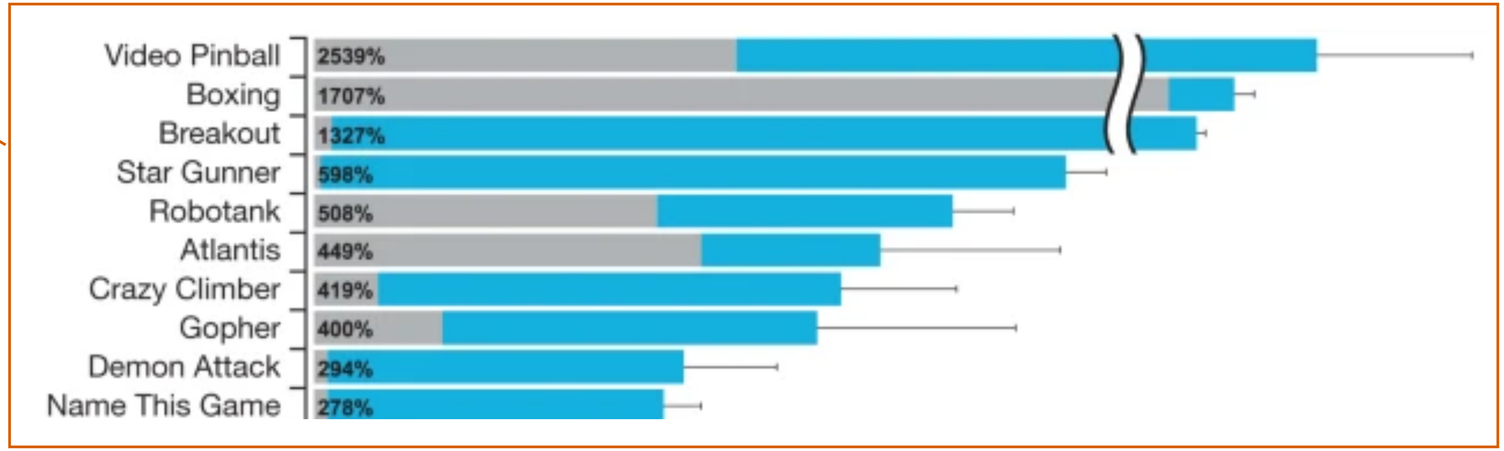
Deep Q-Networks (DQN, Mnih et al. 2015)



4 frames = state

Deep Q-Networks (DQN, Mnih et al. 2015)





Deep Q-Networks (DQN, Mnih et al. 2015)

For episode $= 1, M$ **do**

Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

For $t = 1$,T **do**

With probability ε select a random action a_t

otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$

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 $\left(\phi_t, a_t, r_t, \phi_{t+1}\right)$ in D

Sample random minibatch of transitions $\left(\phi_j, a_j, r_j, \phi_{j+1}\right)$ from D

Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right) & \text{otherwise} \end{cases}$

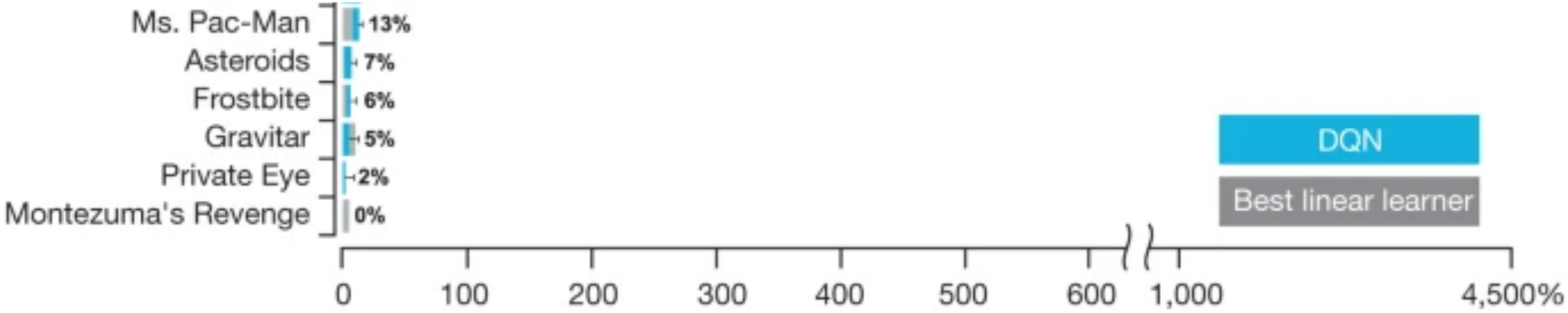
Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ

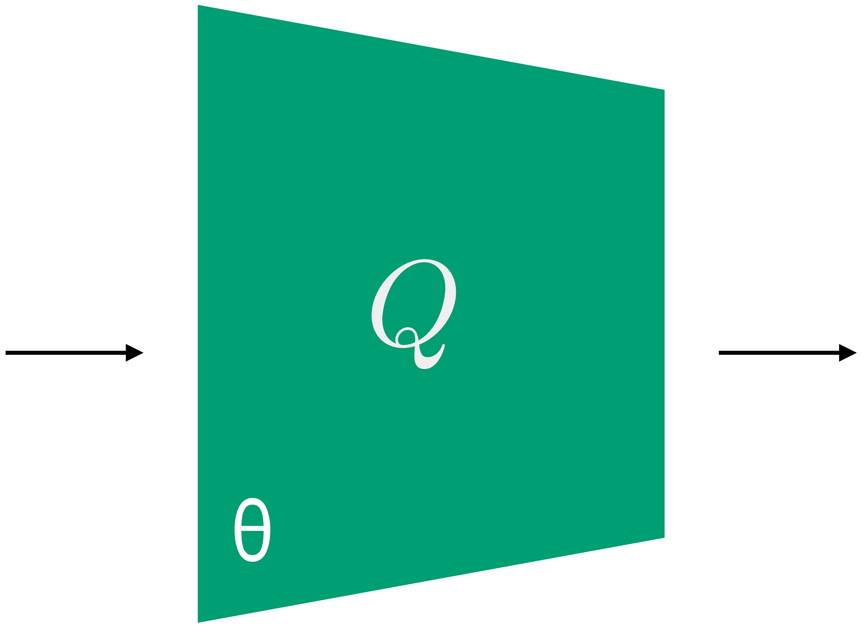
Every C steps reset $\hat{Q} = Q$

Exploration!

Parameter update

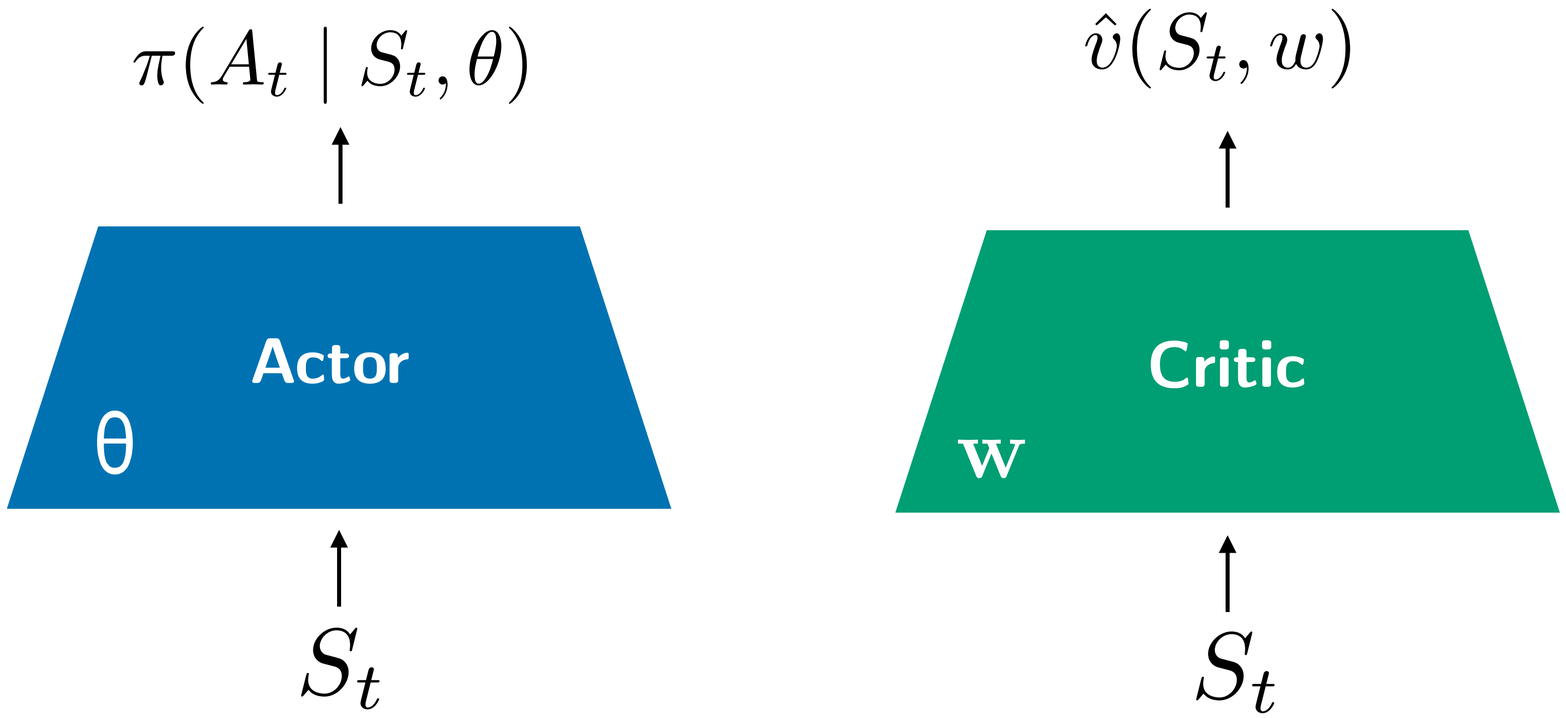
DQN: Limitations



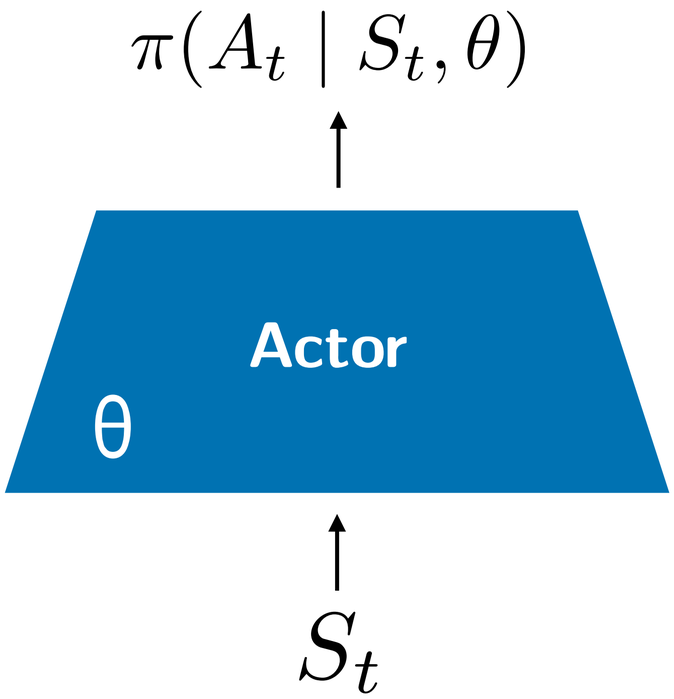


1) Minimal exploration
2) Troubles with delayed, sparse reward
3) Only applicable to discrete action space
4) No convergence guarantees
5) Highly sensitive to hyper parameters

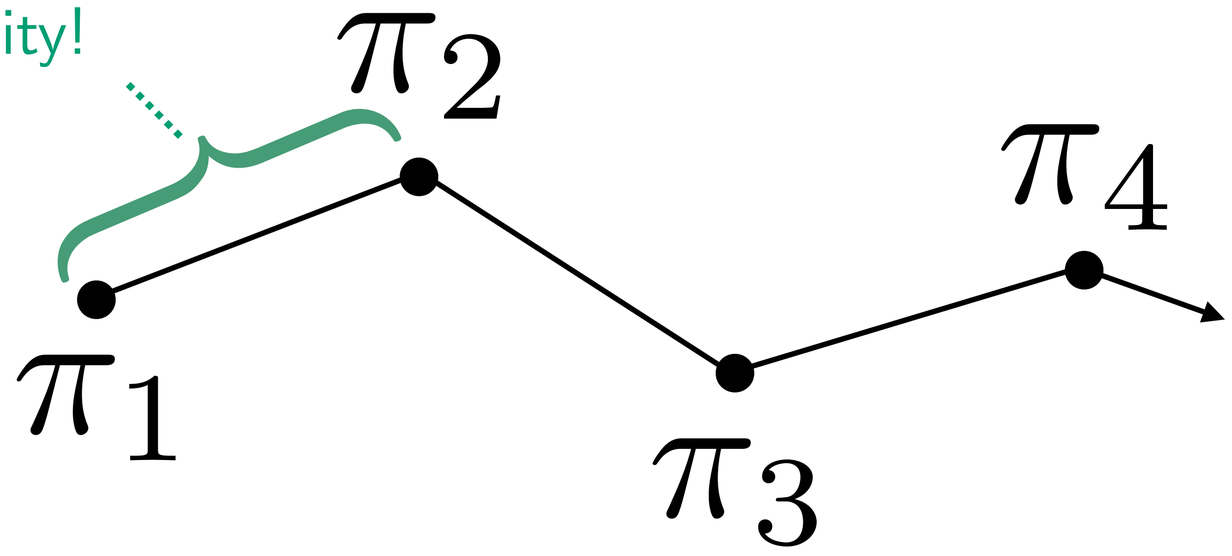
Alg. 2a: Trust Region Policy Optimisation (TRPO, Schulman et al. 2016)



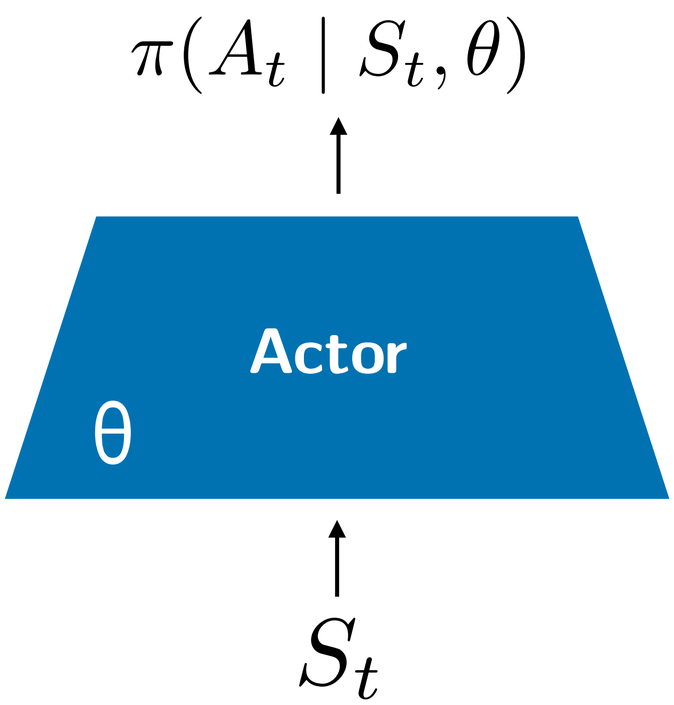
Alg. 2a: Trust Region Policy Optimisation (TRPO, Schulman et al. 2016)

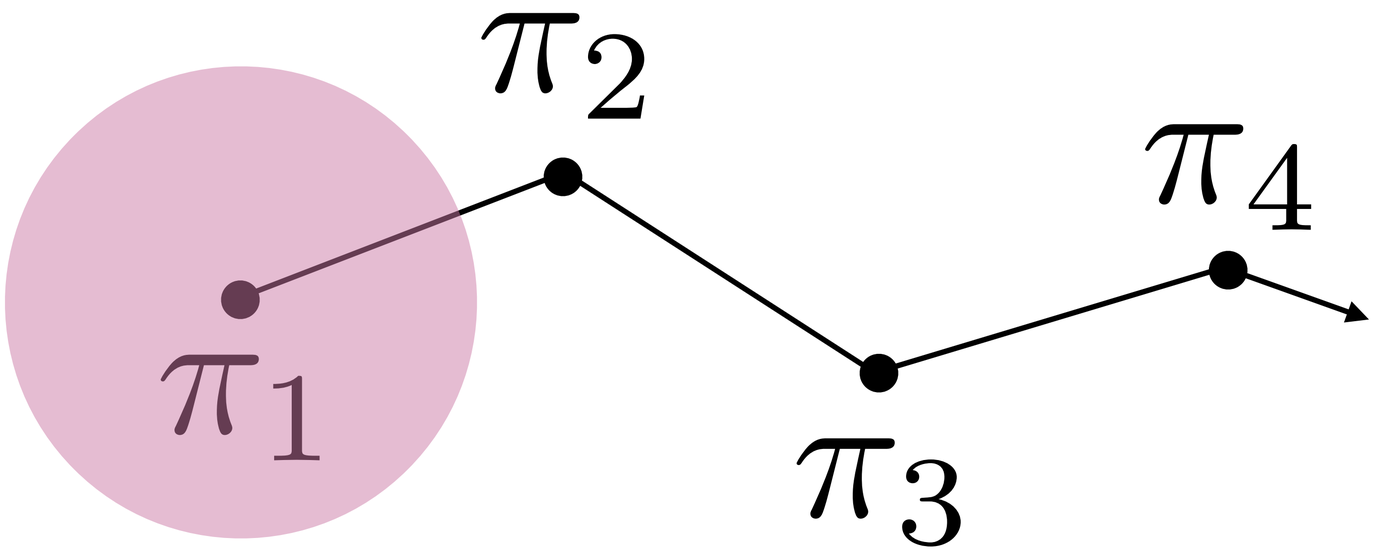


Problem: Big updates yield instability!



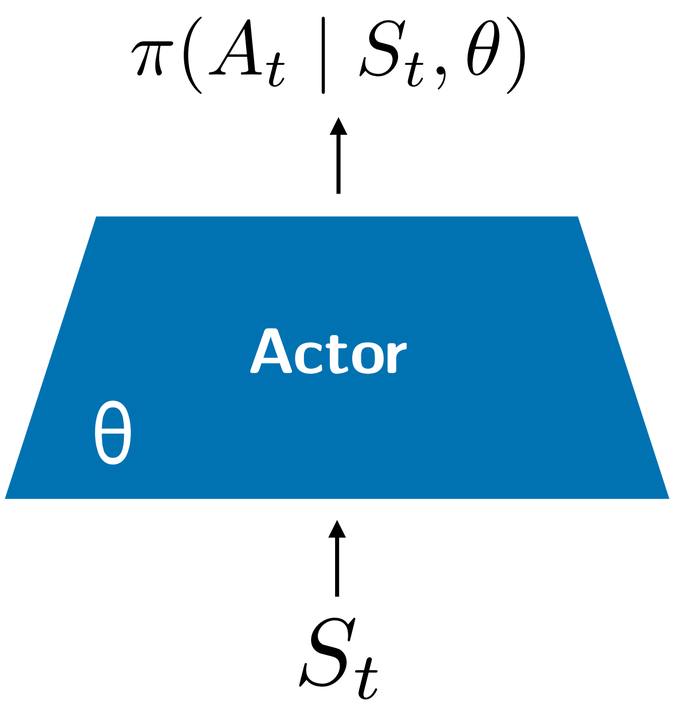
Alg. 2a: Trust Region Policy Optimisation (TRPO, Schulman et al. 2016)

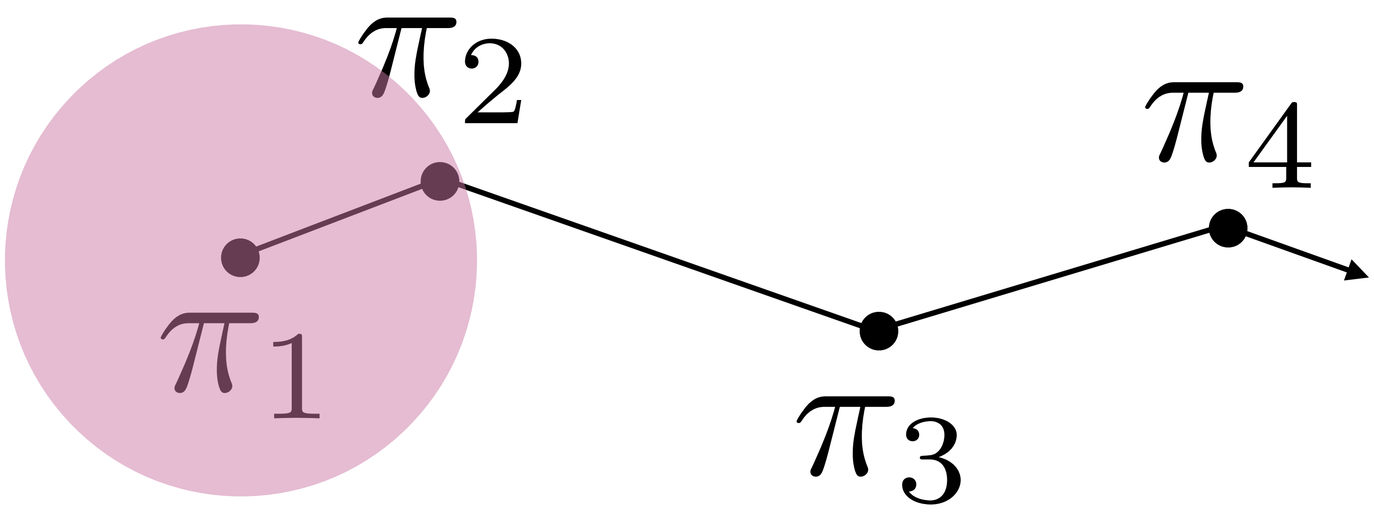




TRPO Main Idea: Keep updates in "Trust Region"

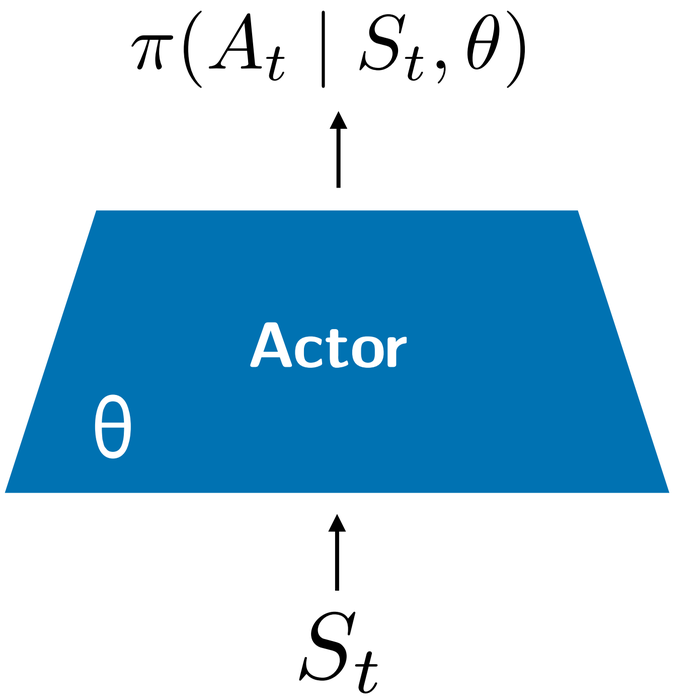
Alg. 2a: Trust Region Policy Optimisation (TRPO, Schulman et al. 2016)

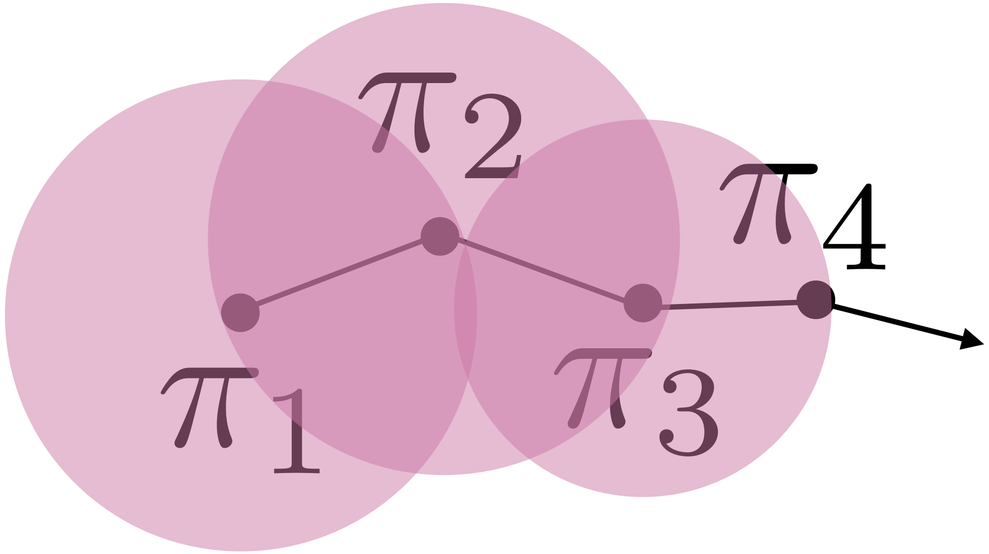




TRPO Main Idea: Keep updates in "Trust Region"

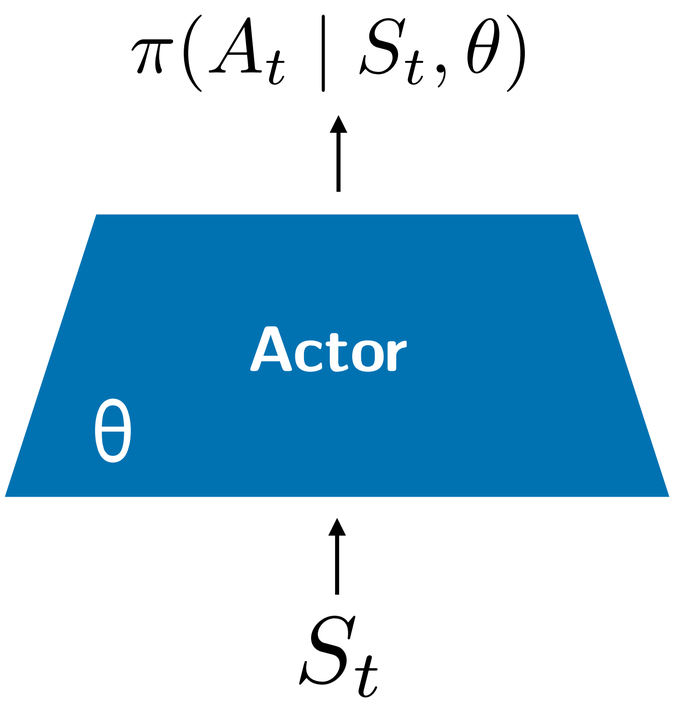
Alg. 2a: Trust Region Policy Optimisation (TRPO, Schulman et al. 2016)





TRPO Main Idea: Keep updates in "Trust Region"

Alg. 2a: Trust Region Policy Optimisation (TRPO, Schulman et al. 2016)



1) Works with continuous action spaces
2) Trust region can yield stability

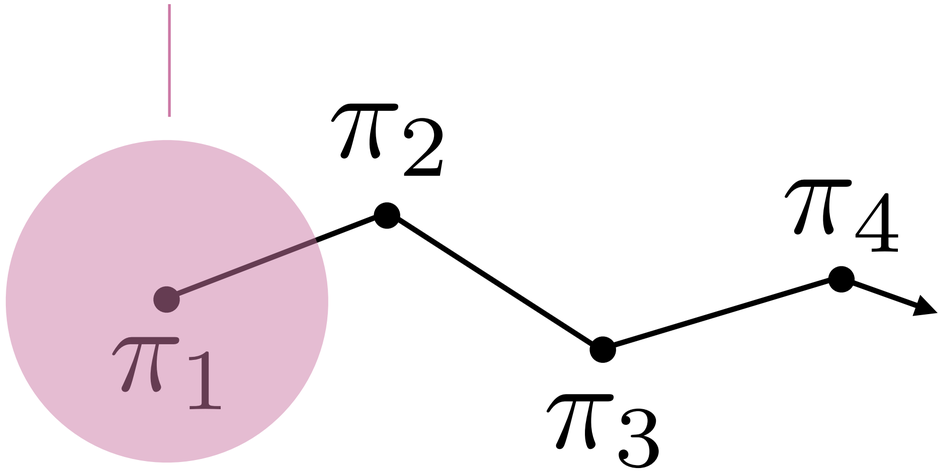
TRPO Advantages

1) Trust region implementation uses KL

TRPO Disadvantage

Alg. 2a: Trust Region Policy Optimisation (TRPO, Schulman et al. 2016)

$$\mathbb{E}_{s\sim\pi_{\theta_t}} \mathrm{D}_{\mathrm{KL}}(\pi_{\theta_{t+1}}(\cdot \mid s) \,||\, \pi_{\theta_t}(\cdot \mid s)) \leq \beta$$

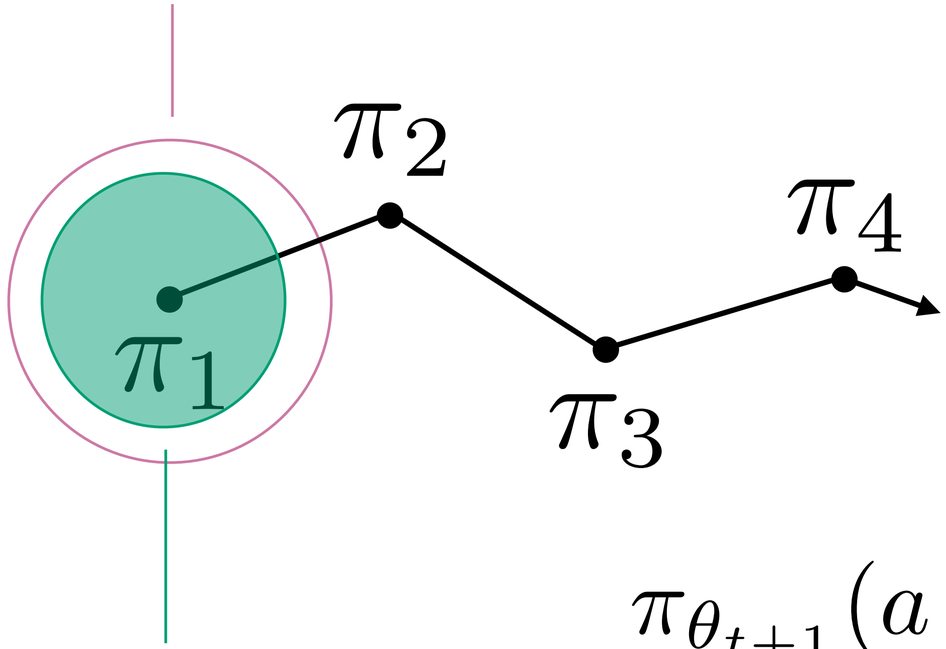


1) Trust region implementation uses KL

TRPO Disadvantage

Alg. 2b: Proximal Policy Optimisation (PPO, Schulman et al. 2016)

$$\mathbb{E}_{s\sim\pi_{\theta_t}} \mathrm{D}_{\mathrm{KL}}(\pi_{\theta_{t+1}}(\cdot \mid s) \,||\, \pi_{\theta_t}(\cdot \mid s)) \leq \beta$$



Uses policy ratio instead: $\dfrac{\pi_{\theta_{t+1}}(a \mid s)}{\pi_{\theta_t}(a \mid s)}$

...for details, see PPO paper

Alg. 2b: Proximal Policy Optimisation (PPO, Schulman et al. 2016)

Algorithm 1 PPO-Clip

1: Input: initial policy parameters θ_0, initial value function parameters ϕ_0
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
4: Compute rewards-to-go $\hat{R}_t$.
5: Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
6: Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \quad g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

typically via stochastic gradient ascent with Adam.
7: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.
8: **end for**

Optional Reading

Hessel et al. (2018)

Haarnoja et al. (2018))

Double Q-learning

Rainbow

DDPG

Soft Actor-Critic

Van Hasselt (2016)

Lillicrap et al. (2018)

Mnih et al.

Schulman et al.

Silver et al.

Schulman et al.