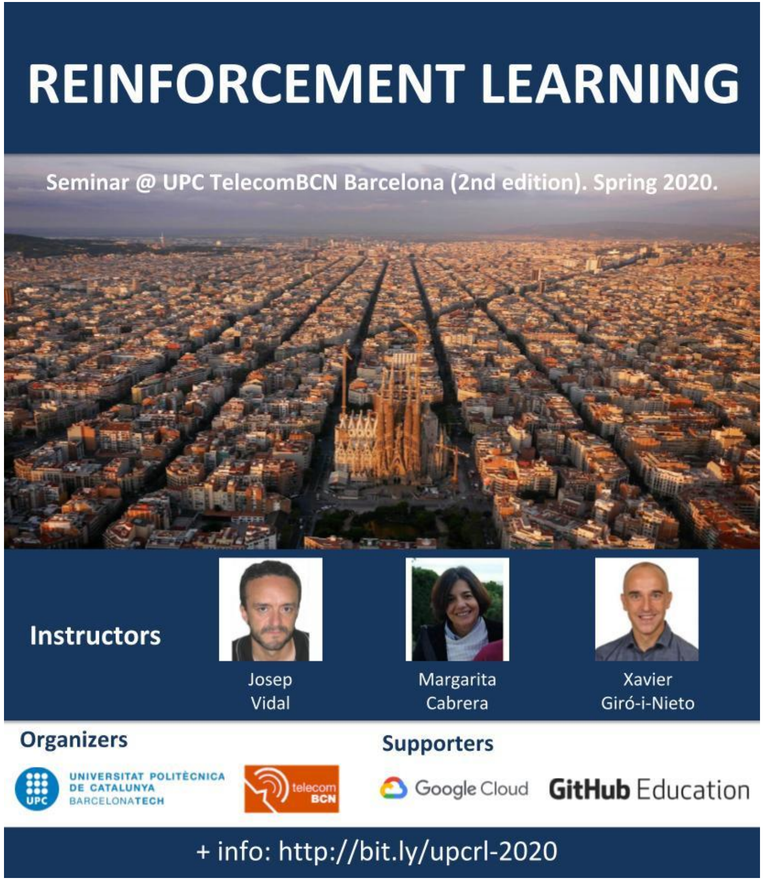


https://telecombcn-dl.github.io/mrl-2020/

MRL 2020 - Day 9 - Part 2

Deep Q-Networks (DQN)

Xavier Giro-i-Nieto

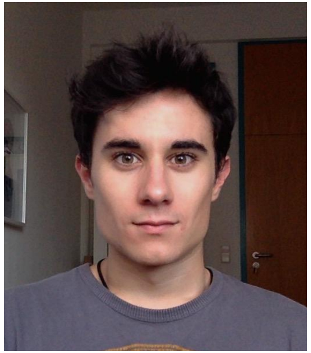
@DocXavi
xavier.giro@upc.edu

Associate Professor
ETSETB TelecomBCN
Universitat Politècnica de Catalunya

UPC

telecos BCN

Acknowledgments



Víctor Campos
victor.campos@bsc.es

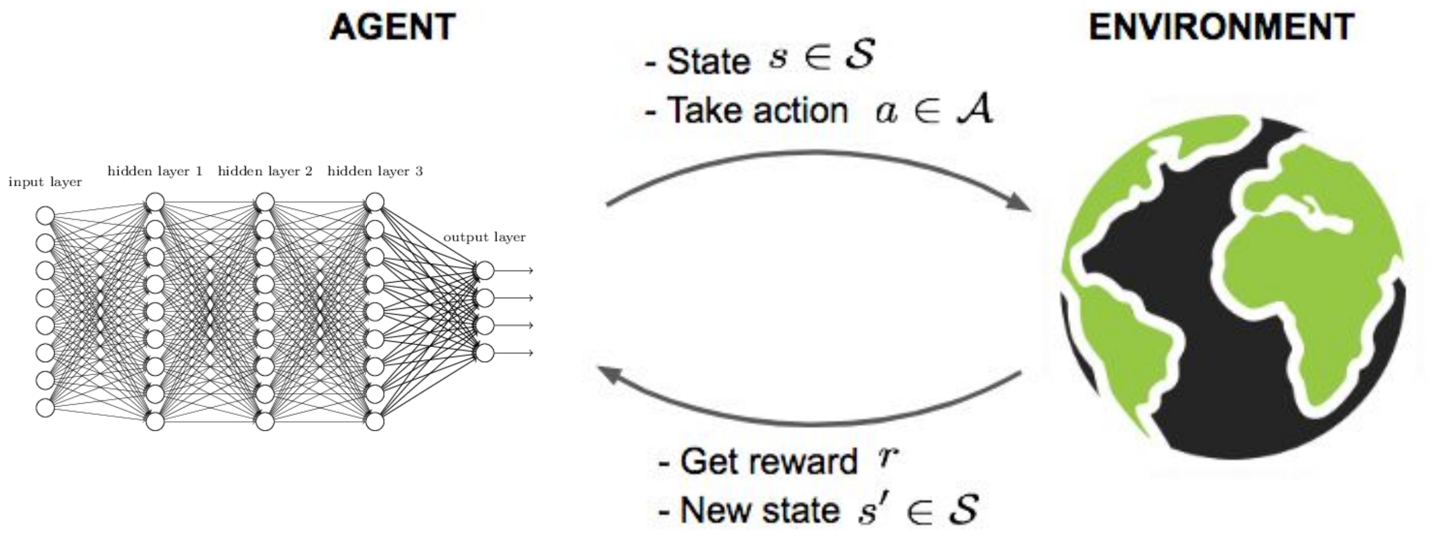
PhD Candidate

Barcelona Supercomputing Center
Universitat Politècnica de Catalunya

Outline

1. **Motivation**
2. Q-Learning with Neural Networks
3. Deep Q-Networks (DQN)

Reinforcement Learning with Neural Networks (NN)



Policy π

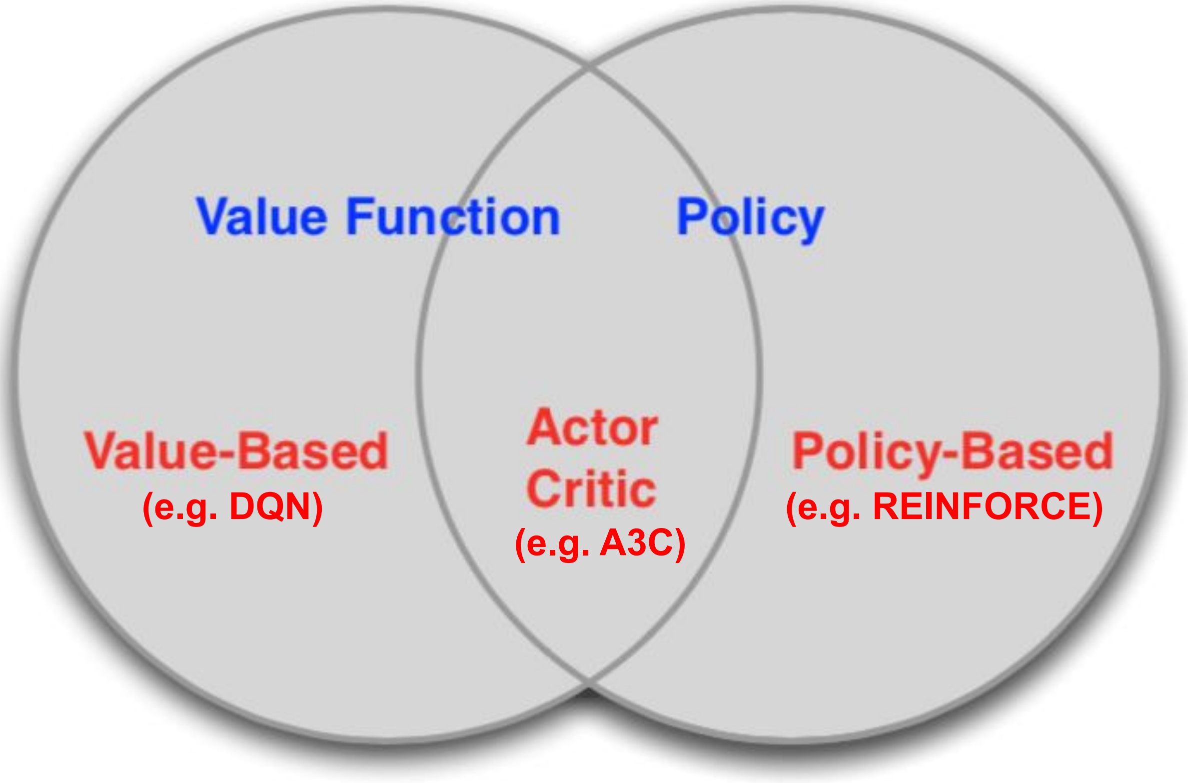
Value function

Model
(of the Environment)

Goals of Reinforcement Learning

Flavours of (model-free) RL

Policy-based / value-based



Who generates the training data ?

On-policy / off-policy

$$\pi^* = \arg\max_{\pi} \underline{\mathbb{E}_{\tau \sim \mathcal{M},\pi}}[R_\tau]$$

$$V_\pi(s) = \underline{\mathbb{E}_\pi}[G_t | S_t = s]$$

$$Q_\pi(s, a) = \underline{\mathbb{E}_\pi}[G_t | S_t = s, A_t = a]$$

The underlined parts are <u>very</u> important!

Who generates the training data ?

On-policy / off-policy

We need to collect data by following (i.e. running) the current policy.

This has some implications:

- Once we perform an update, we can’t use that same data anymore: we need to *create a new dataset every time*.
- Combined with the slow convergence of SGD (we use small learning rates!), this results in data-inefficient methods.
- We can’t use expert demonstrations

Who generates the training data ?

On-policy / off-policy

Can we do something about this? Yes: **off-policy learning** (e.g. Q-learning). But off-policy learning is not easy!

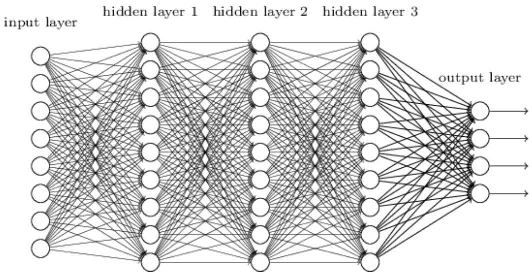
- Maths become more complex
- Learning can be unstable
- We still need some overlap or similarity between the target policy providing the data and the behaviour policy we are trying to learn.

Who generates the training data ?

On-policy / off-policy

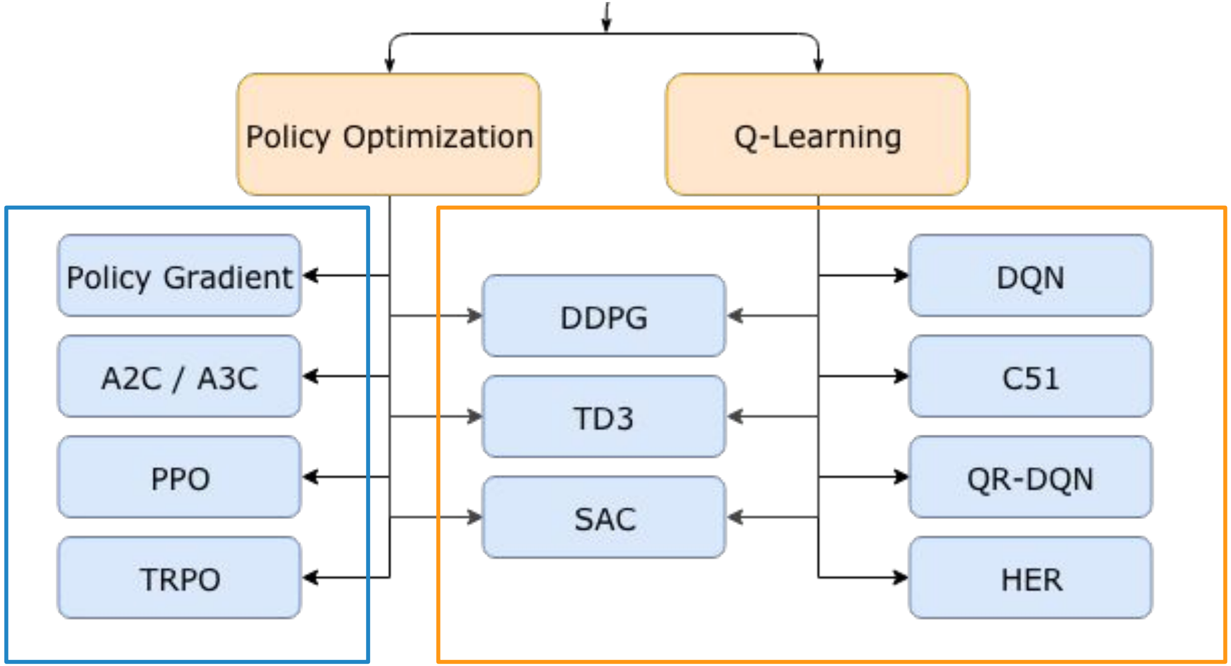
Despite these problems, the data efficiency of off-policy learning is much higher than on-policy learning.

Off-policy RL can reuse the same samples many times -- which is a must for efficient training of neural networks with SGD.



Who generates the training data ?

On-policy / off-policy



Policy gradient methods
(on-policy)

Extensions of DQN
(off-policy)

Outline

1. Motivation
2. **Q-Learning with Neural Networks**
3. Deep Q-Networks (DQN)

Solving the Optimal Policy

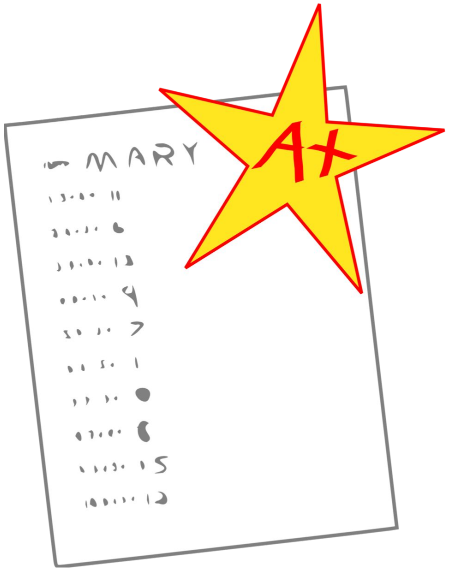
The **optimal policy** is that one capable of achieving the optimal value functions $V_*(s)$ and $Q_*(s,a)$

Optimal policy π_*

$$\pi_* = \arg\max_{\pi} Q_\pi(s, a)$$

Optimal Q-value functions

$$Q_*(s, a) = \max_{\pi} Q_\pi(s, a)$$

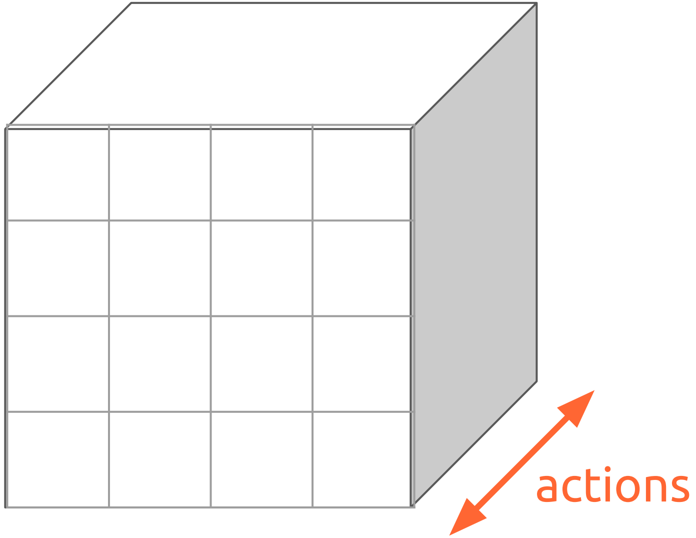


Solving the Optimal Policy: Q-Learning

Tabular Q-Learning is feasible for small state-action spaces:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a \in \mathcal{A}} Q(S_{t+1}, a) - Q(S_t, A_t)).$$

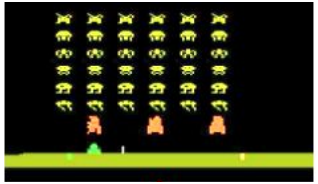
Q(s,a)



Q-Learning with Neural Networks

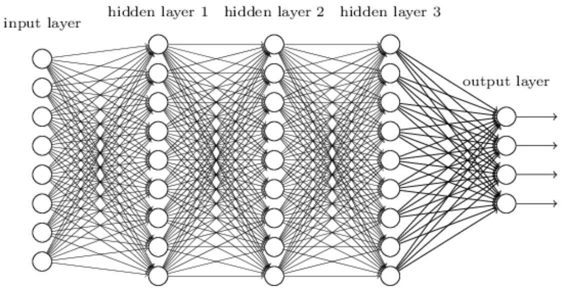
Exploring all positive states and action is **not scalable**

Eg. If video game, it would require generating all possible pixels and actions.



$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a \in \mathcal{A}} Q(S_{t+1}, a) - Q(S_t, A_t)).$$

Solution: Use a neural network as an **function approximator** of Q*(s,a).

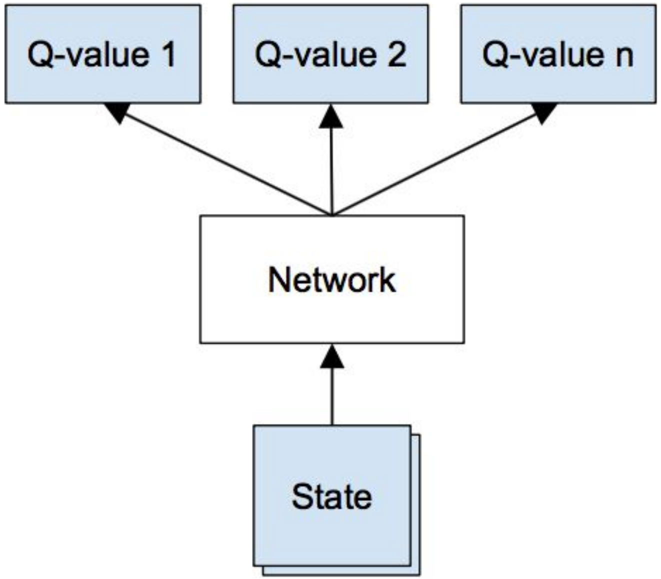


$$\boxed{Q(s,a,\Theta) \approx Q^*(s,a)}$$

Neural Network parameters

Q-Learning with Neural Networks

Value-based methods maintain an estimate of a value function (Q, V and/or A), but not a policy. The policy is implicit:

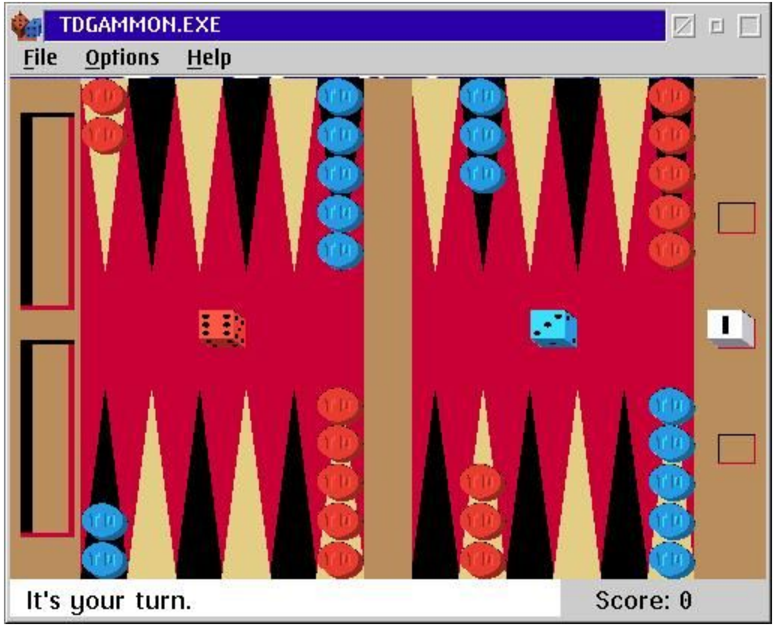


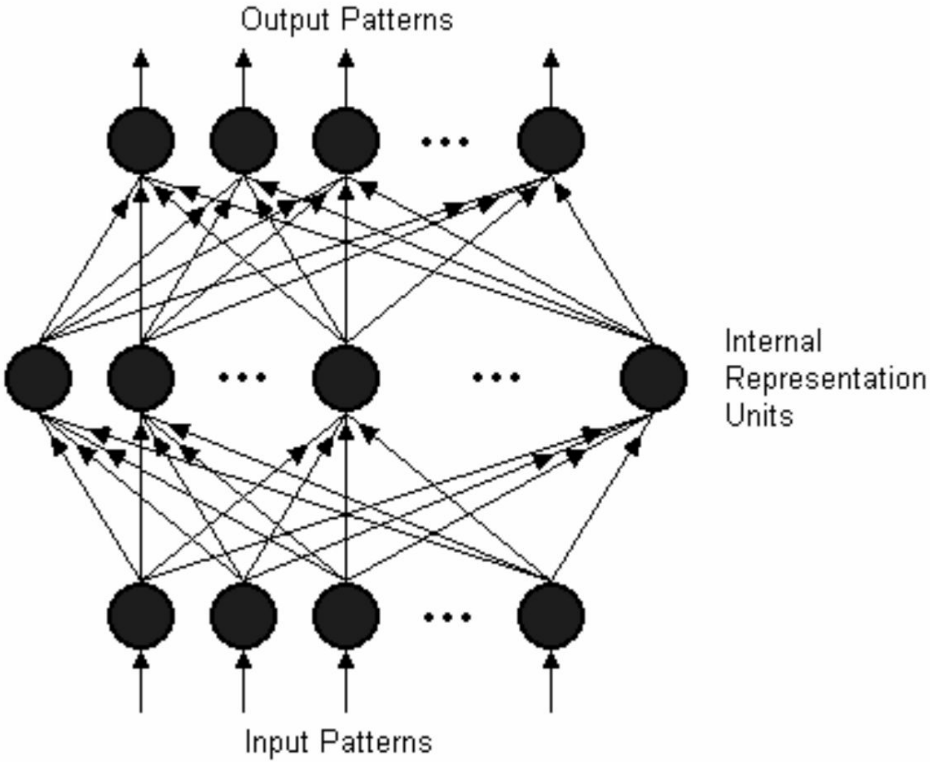
ε-Greedy policy

prob(ε) → $\pi(s) = rand(A)$

prob(1-ε) → $\pi(s) = \arg\max_{a \in A} Q(s,a)$

Q-Learning with Neural Networks





#TD-Gammon Tesauro, G. (1994). [TD-Gammon, a self-teaching backgammon program, achieves master-level play. Neural computation](), 6(2), 215-219.

Outline

1. Motivation
2. Q-Learning with Neural Networks
3. **Deep Q-Networks (DQN)**

Deep Q-learning (DQN)

Deep Q-Network (DQN) was the first method to combine value-based RL (in particular, Q-learning) with deep neural networks.

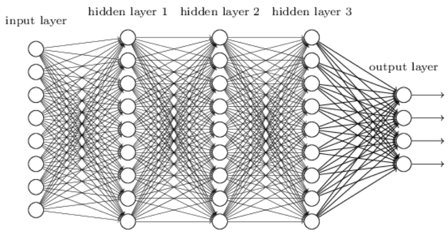
Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves et al. "Human-level control through deep reinforcement learning." *Nature* 518, no. 7540 (2015): 529-533.

DQN: Policy & Target Networks

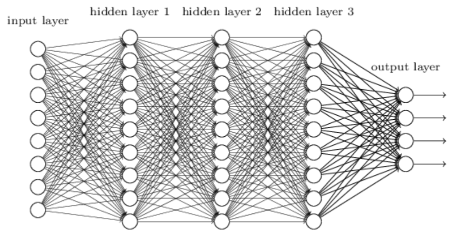
- Q-network parameters determine the next training samples ➡️ can lead to bad feedback loops .

- A separate and more stable **target network** is used to estimate TD targets.

Target Network (w_i^-)



Policy Network (w_i)



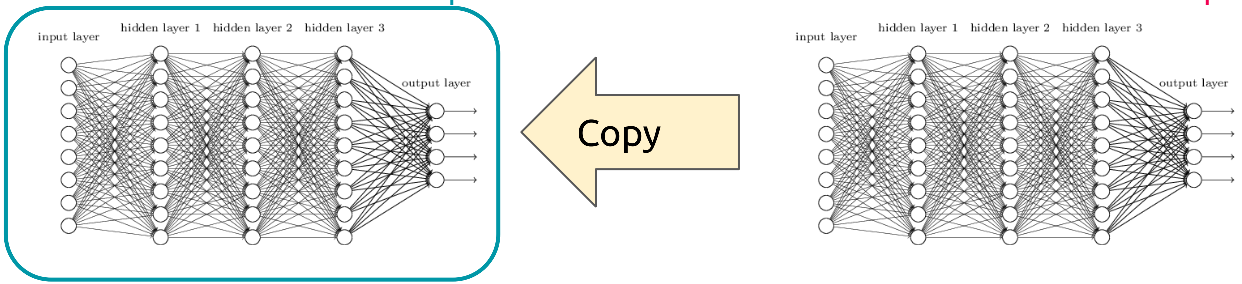
Source: Arthur Juliani, [“Simple Reinforcement Learning with Tensorflow Part 4: Deep Q-Networks and Beyond”] (2016)

DQN: Policy & Target Networks

The target network (w_i^-) is updated by copying the parameter of the policy network (w_i) periodically.

Target Network (w_i^-)

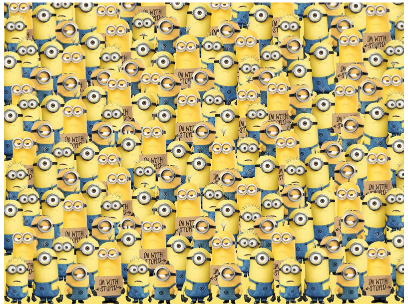
Policy Network (w_i)



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

Source: Arthur Juliani, "Simple Reinforcement Learning with Tensorflow Part 4: Deep Q-Networks and Beyond" (2016)

DQN: Replay Memory



- Learning from batches of **consecutive samples** is problematic because samples are too correlated ➡️ inefficient learning

- Continually update a **replay memory** table of transitions (s_t, a_t, r_t, s_{t+1}) as episodes are collected.
- Train a the policy network (w_i) with **random minibatches** of transitions from the replay memory, instead of consecutive samples.

Memory			
s_t	a_t	r_{t+1}	s_{t+1}
s_t	a_t	r_{t+1}	s_{t+1}
s_t	a_t	r_{t+1}	s_{t+1}
s_t	a_t	r_{t+1}	s_{t+1}
…	…	…	…
s_t	a_t	r_{t+1}	s_{t+1}

Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

Deep Q-learning (DQN)

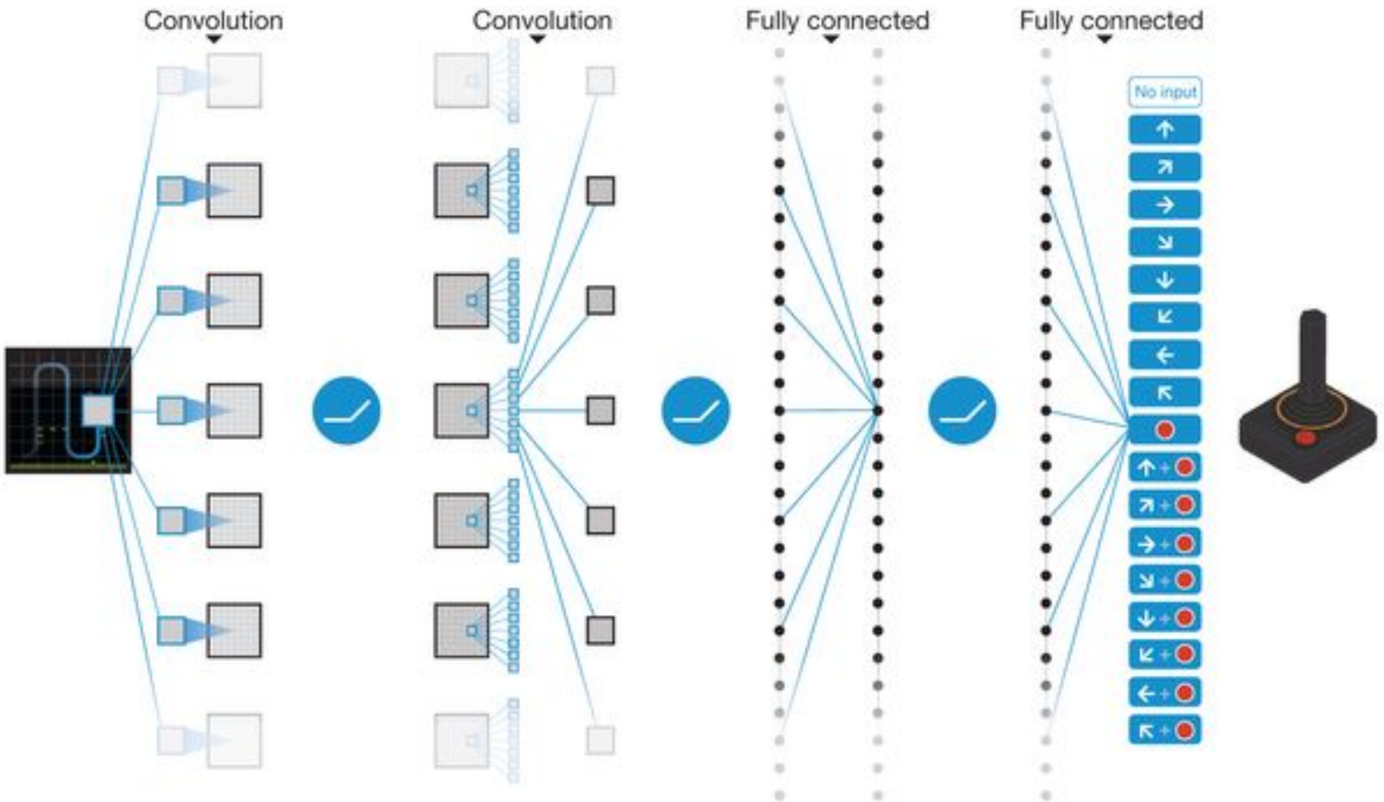
Algorithm:

1. Collect transitions (s_t, a_t, r_{t+1}, s_{t+1}) and store them in a **replay memory** *D*
2. Sample random mini-batch of transitions (s, a, r, s') from **replay memory** *D*
3. Compute TD-learning targets wrt old parameters w^-
4. Optimise with MSE loss using gradient descent:

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

David Silver, UCL course on RL (Lecture 6)

Deep Q-learning (DQN)



Number of actions between 4-18, depending on the Atari game

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves et al. "Human-level control through deep reinforcement learning." *Nature* 518, no. 7540 (2015): 529-533.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).

Deep Q-learning (DQN)

Artificial intelligence (AI)

Google buys UK artificial intelligence startup Deepmind for £400m

Google makes its biggest EU purchase yet with the technology that aims to make computers think like humans

Samuel Gibbs

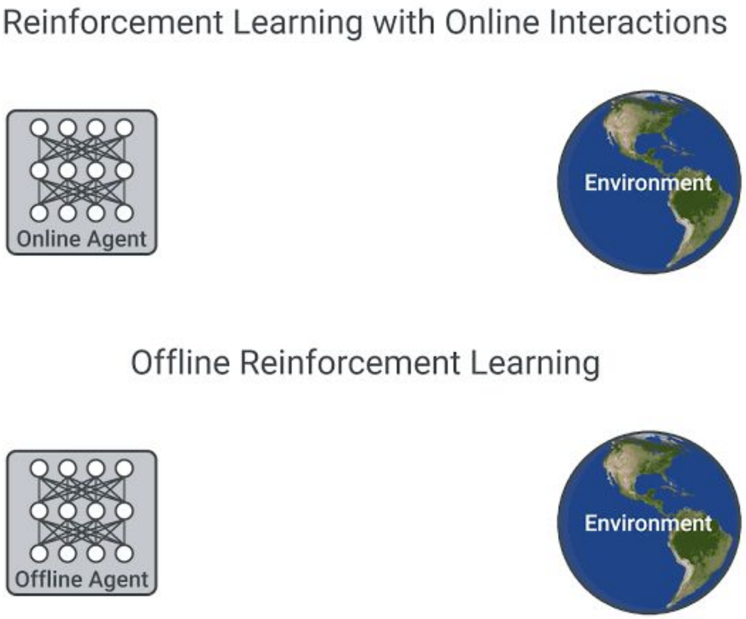
Monday 27 January 2014
13.23 GMT

This article is **2 years** old

1046 186

Fully Off-policy RL = Batch RL = Offline RL

DQN agents have also learned to play Atari games by only watching DQN's replay.

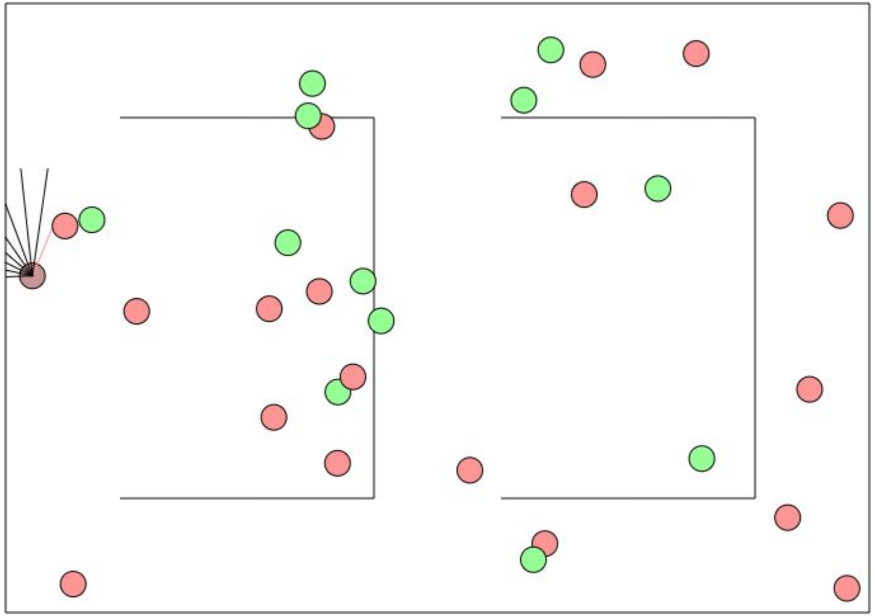


#DQN-Replay-Dataset Rishabh Agarwal, and Mohammad Norouzi, "An Optimistic Perspective on Offline Reinforcement Learning", Google AI (2020)

Online demo

ConvNetJS

Deep Learning in your browser



Andrej Karpathy, [“ConvNetJS Deep Q Learning Demo”]

Outline

1. Motivation
2. Q-Learning with Neural Networks
3. Deep Q-Networks (DQN)

Learn more

Kamal Ndousse, "DQN and DRQN in partially observable gridworlds" (2020)

Sergey Nikolenko, "Deep Q-Network" (includes TF-Gammon) (2017)