Deep Q Learning

Prof. Alfio Ferrara

Reinforcement Learning

Deep Q Learning architecture

Deep Q Learning does not solve the problem of convergence to the optimal policy, but has been proven to work in many cases. Deep Q Learning is an algorithm for control, so it is based on the following approximation:

$$\hat{Q}(s, a, \mathbf{w}) \approx Q^{\pi}(s, a) \tag{1}$$

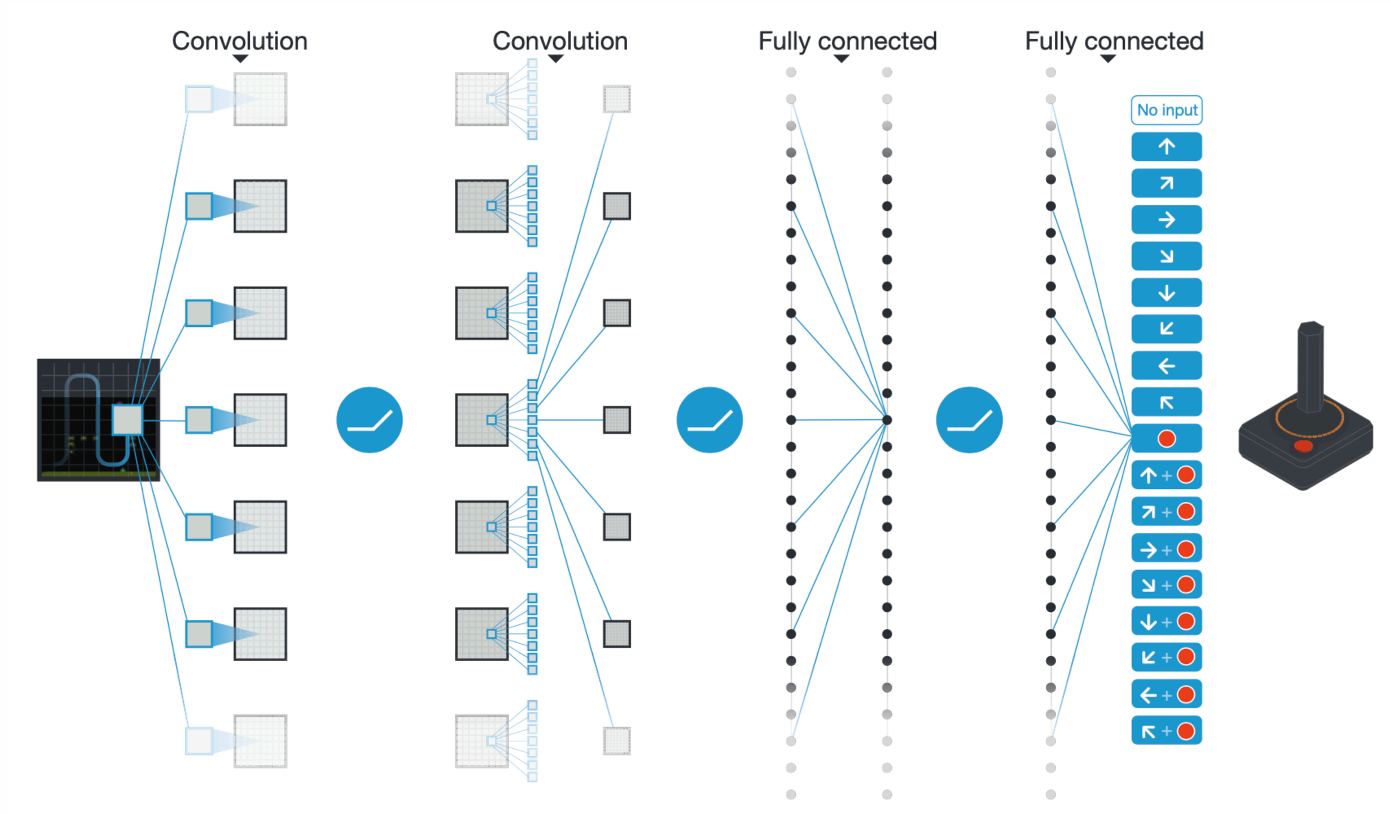
Let's recap the update equation for Q Learning

$$\Delta_{\mathbf{w}} = \alpha \left(r + \gamma \max_{a} \left(\hat{Q}(s_{t+1}, a, \mathbf{w}) - \hat{Q}(s_t, a_t, \mathbf{w}) \right) \right) \nabla_{\mathbf{w}} \hat{Q}(s_t, a_t, \mathbf{w}) \tag{2}$$

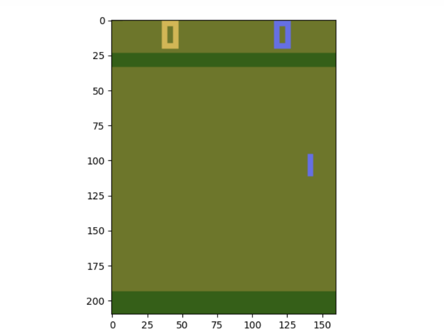
Deep Q Learning was original proposed to solve Atari games.

> Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *nature*, *518*(7540), 529-533.

Atari input is seen as an image having each pixel configuration as a state (i.e., $84 \times 84 \times 4$ Input values) and a DNN with three convolutional layers and two fully connected layers (image from the paper).



Note: a nice environment for Atari is available on `gymnasium`. See for example Pong.



The way DQN paper addresses the problem of convergence of Q learning is by introducing two mechanisms, *experience replay* and *fixed Q-targets*

Experience replay

A *replay buffer* $\mathcal{D}$ is used to collect observations from previous play in form

$$\begin{gathered}(s_t, s_t, r_t, s_{t+1}),\\ (s_{t+1}, s_{t+1},\\ r_{t+1}, s_{t+1},\\ \dots,\\ (s_{T-1}, s_{T-1}, r_{T-1}, s_T)\end{gathered} \tag{3}$$

Then, using $\mathcal{D}$, perform the following

- Sample (s_i, a_i, r_i, s_{i+1}) from $\mathcal{D}$

- Compute the target value $r + \gamma \max_{a_{i+1}} \hat{Q}(s_{i+1}, a_{i+1}, \mathbf{w})$
- Use this target to update the weights for the sample

The point here is that the estimation of $\hat{Q}$ changes over time, but it is update of previous experienced games in order to limit the chance to get a divergent value for always new observations.

Fixed Q Targets

The idea is that, instad of computing the target using the continuously updated weights $\mathbf{w}$, we keep a second set of *fixed weights* $\mathbf{w}'$. We use $\mathbf{w}'$ to compute the target, but we update the weights $\mathbf{w}$. This way we reduce the risk of divergence due to a continuous update of weights.

The new way we update weights is now

$$\Delta_{\mathbf{w}} = \alpha \left(r + \gamma \max_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1}, \mathbf{w}') - \hat{Q}(s_t, s_t, \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{Q}(s_t, s_t, \mathbf{w}) \tag{4}$$

The fixed ways can be updated but in general, we keep them for several rounds because we do not want $\mathbf{w}$ to explode towards infinity or zero. Every k observation, we perform $\mathbf{w}' \leftarrow \mathbf{w}$.

Both Experience replay and Fixed Q targets have the goal to improve the system stability during the learning process.