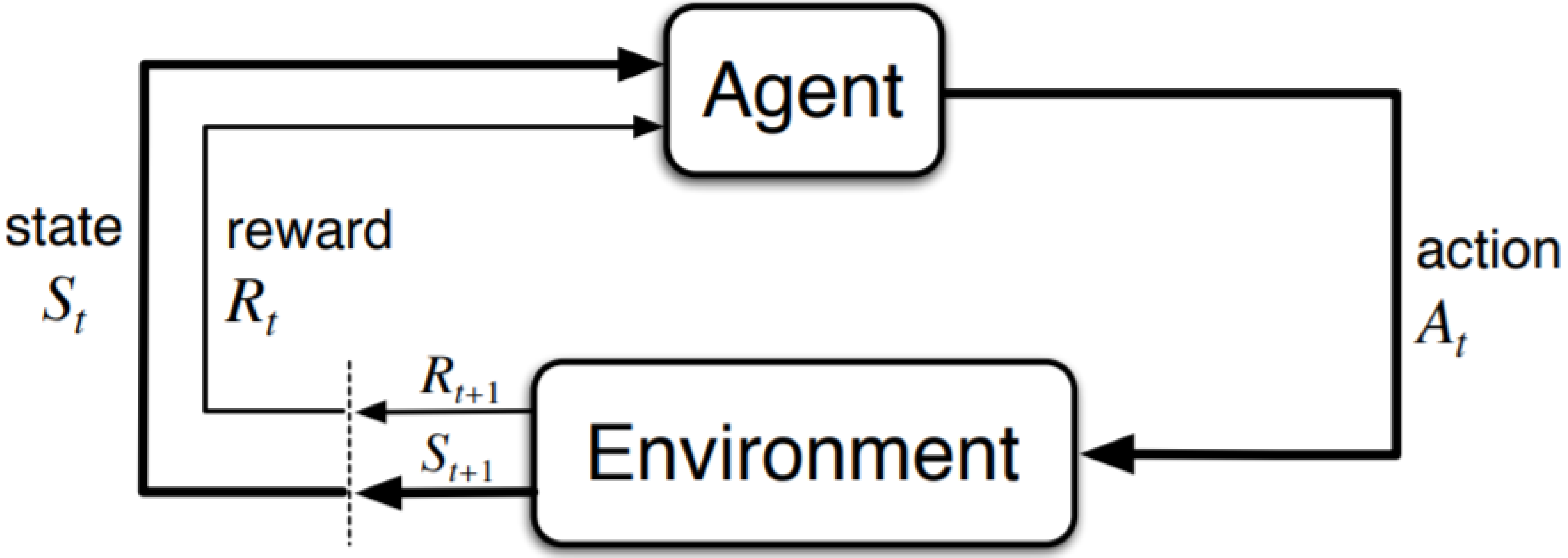
Deep Q-Network (DQN)

by Alina Vereshchaka

Finite Markov Decision Processes (MDP)



Episode: $s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3, a_3, \ldots, s_T$

Notations Recap

Return: $$G_t = R_{t+1} + \gamma R_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

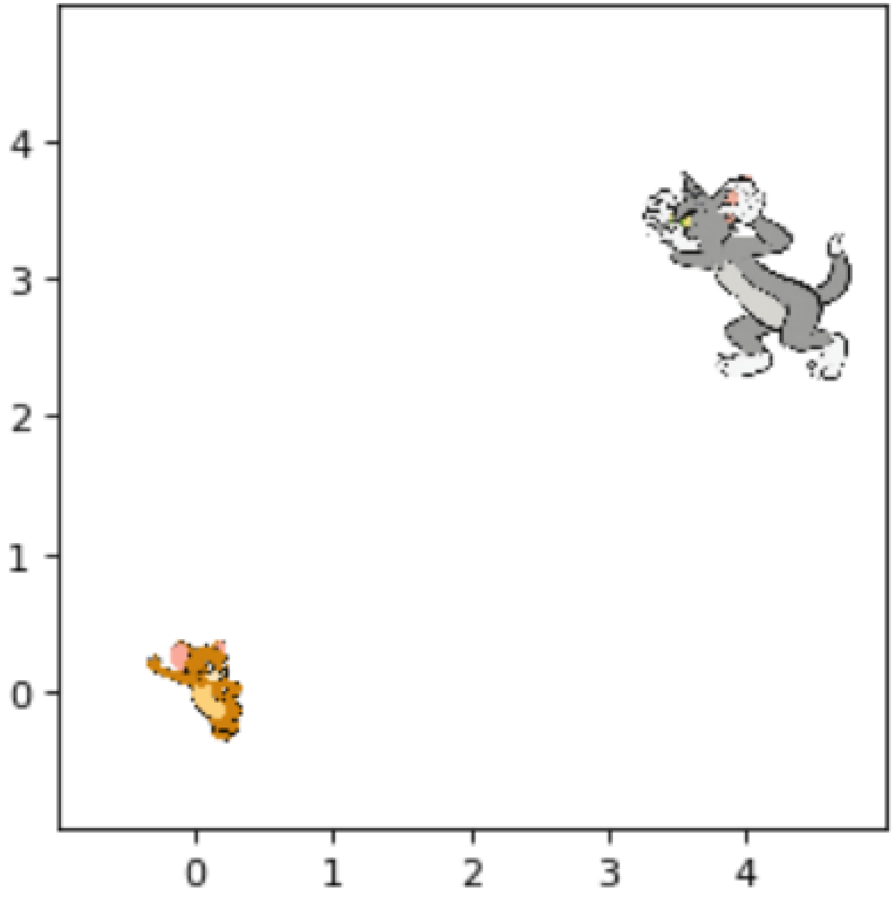
Q-value function: $$Q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$$
$$= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right]$$

Deterministic policy: $\pi(s) = a$

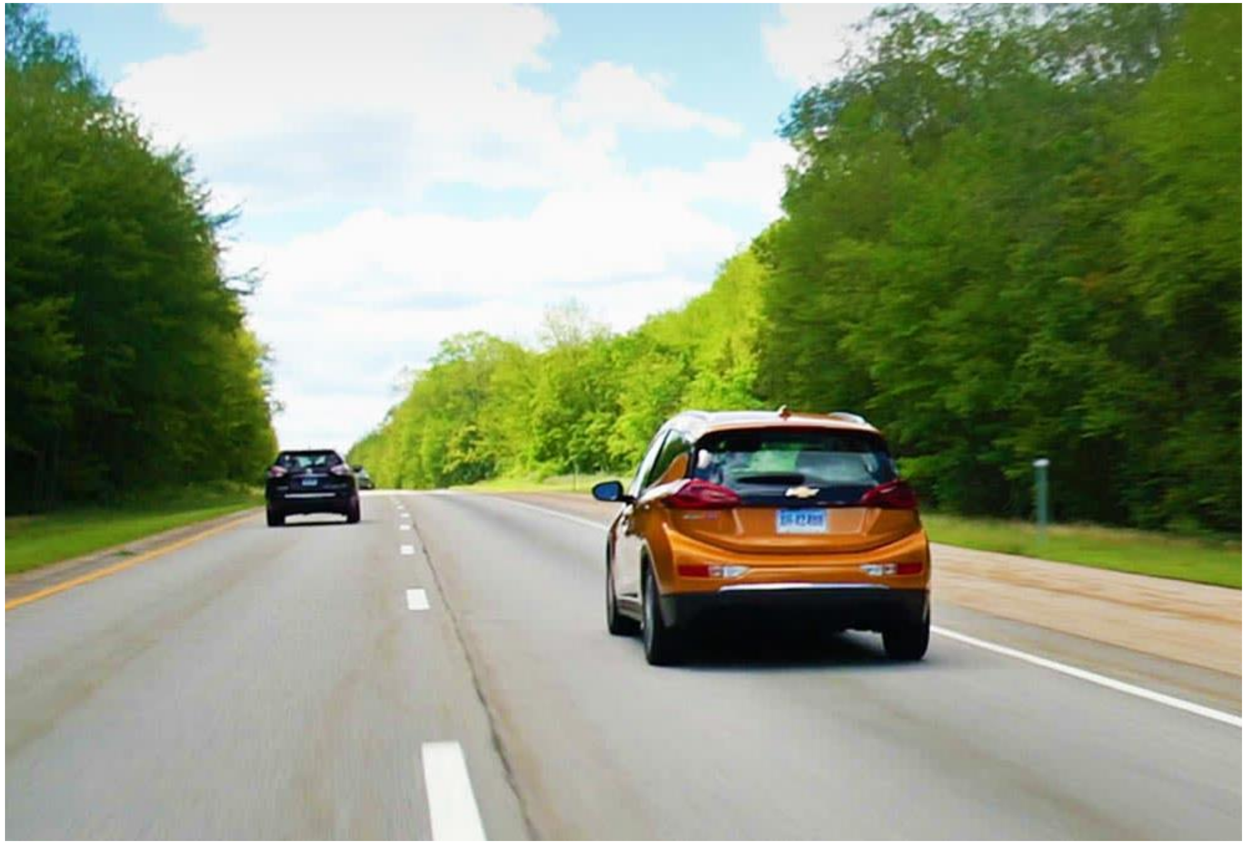
Objective: $\boxed{\pi^*(s) = argmax_a Q(s,a)}$

Environments

Fully observable



Partially observable



Deep Reinforcement Learning: AI = RL + DL

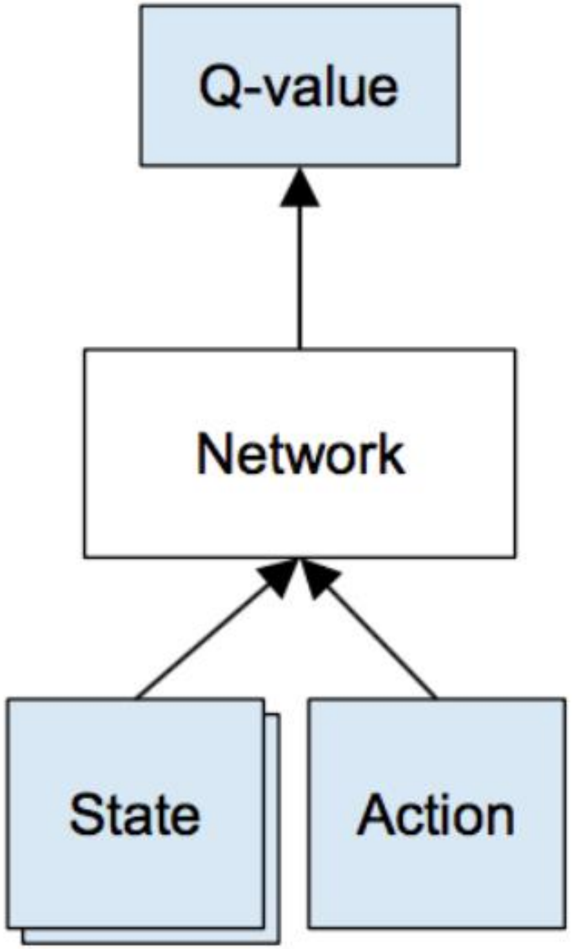
We seek a single agent which can solve any human-level task

- Reinforcement Learning(RL) defines the **objective**
- Deep Learning(DL) gives the **mechanism**

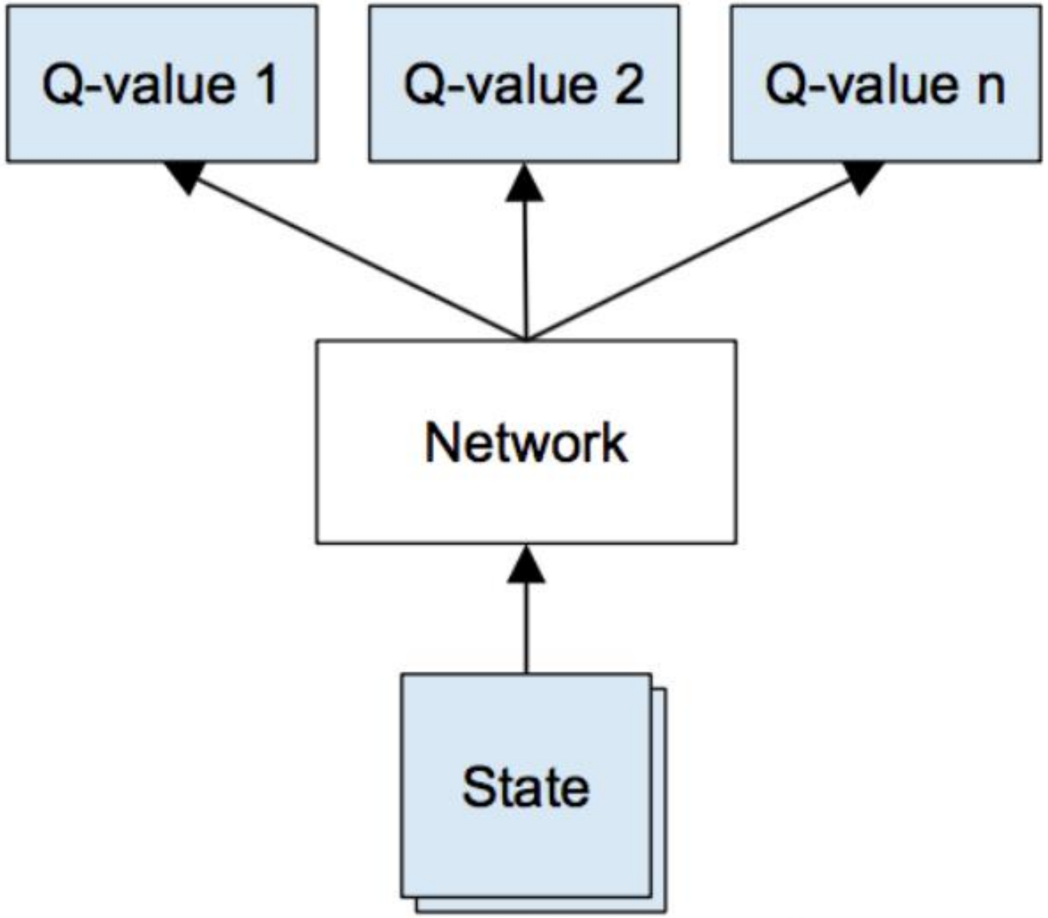
RL + DL = general intelligence

DQN Architectures

Naive DQN



Optimized DQN, used by DeepMind



DQN Algorithm (modified from DeepMinds)

Initialize replay memory to capacity N

Initialize the environment (reset)

For episode $= 1, M$ **do** (Begin a loop of interactions between the agent and environment)

Initialize the first state $s_0 = s$

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

With probability ϵ select a random action a_t, otherwise select $a_t = argmax_a Q(s, a; \Theta)$

Execute action a_t and observe reward r_t and next state s_{t+1}

A tuple $< s_t, a_t, r_t, s_{t+1} >$ has to be stored in memory

Sample random minibatch of observations (s_t, a_t, r_t, s_{t+1}) from memory

Calculate Q-value

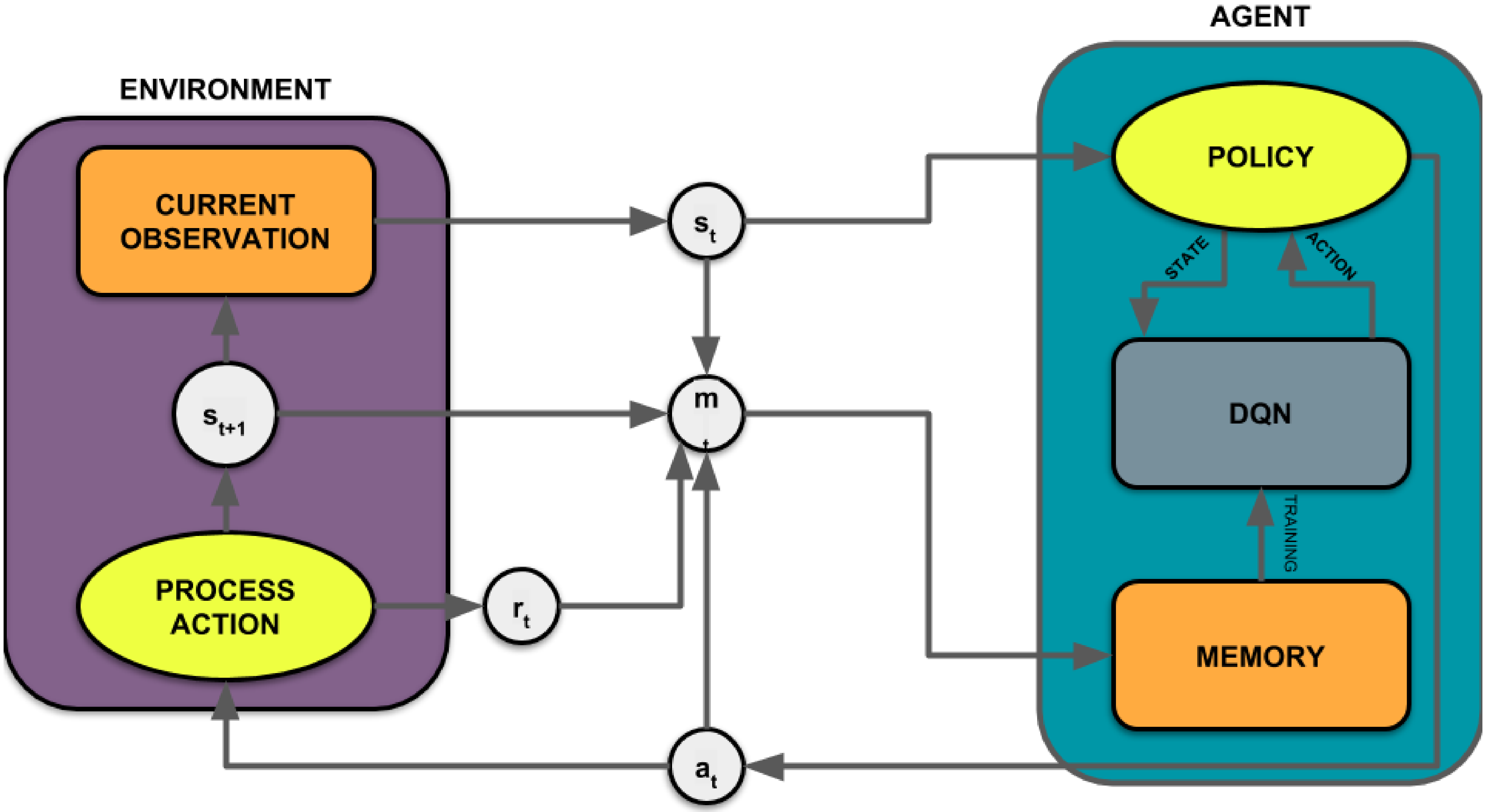
$$Q_t = \begin{cases} r_t, & \text{if episode terminates at step } t+1 \\ r_t + \gamma max_a Q(s_t, a_t; \Theta), & \text{otherwise} \end{cases}$$

Train a neural network on a sampled batched from the memory

End **For**

End **For**

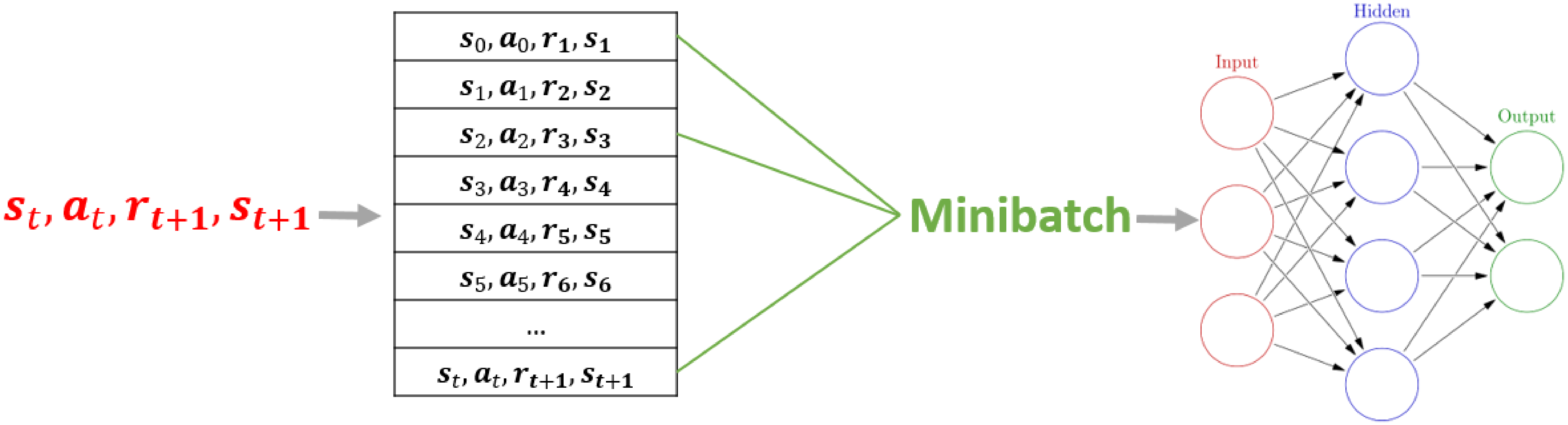
Deep Q-Learning Process



Experience Replay

Problem: approximation of Q-values using non-linear functions is not stable

Solution:



Exploration vs Exploitation

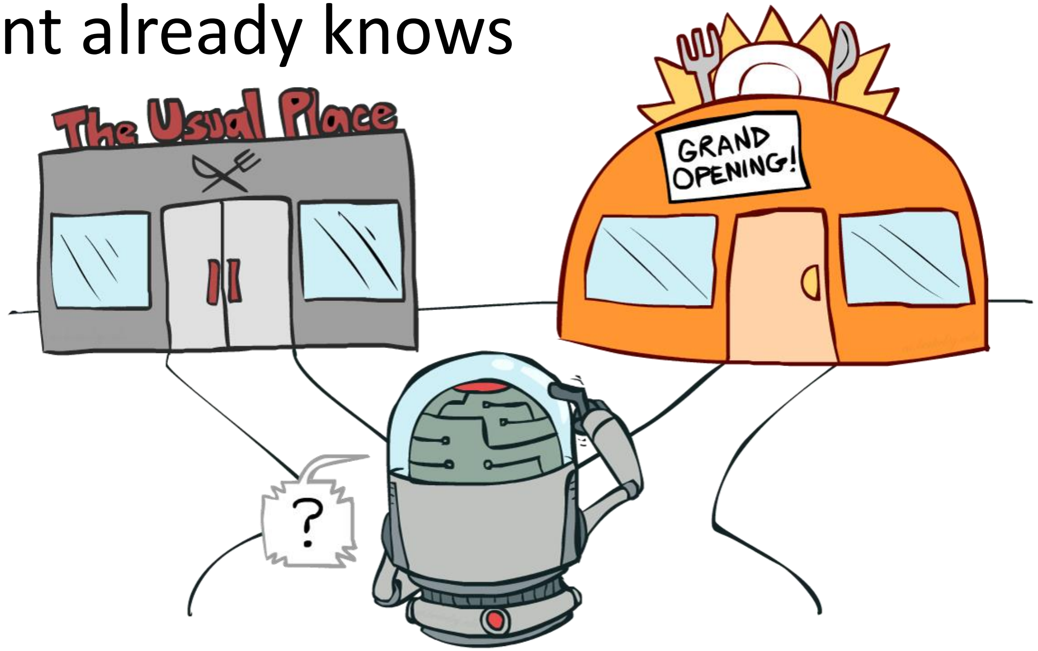
Exploration:

- Discover better action selections
- Improve the knowledge about the environment

Exploitation:

- Maximize the reward based on what agent already knows

Exploration-exploitation dilemma: both can't be pursued exclusively without failing



DQN: Atari

Environment: BreakoutDeterministic-v4

Backend: Keras, Python3

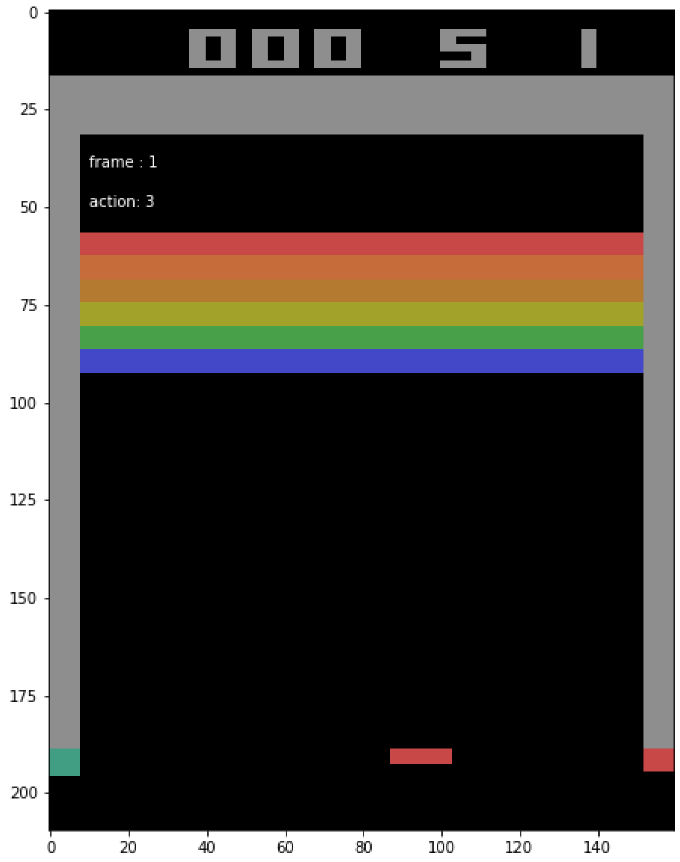
Libraries: OpenAI Gym, Keras-RL

Model: CNN

Reward: max score - 208 (benchmark is 225)

Preprocessing: original image was downsampled from 210×160 pixel images to 105×80 and converted from RGB to gray-scal to decrease the computation

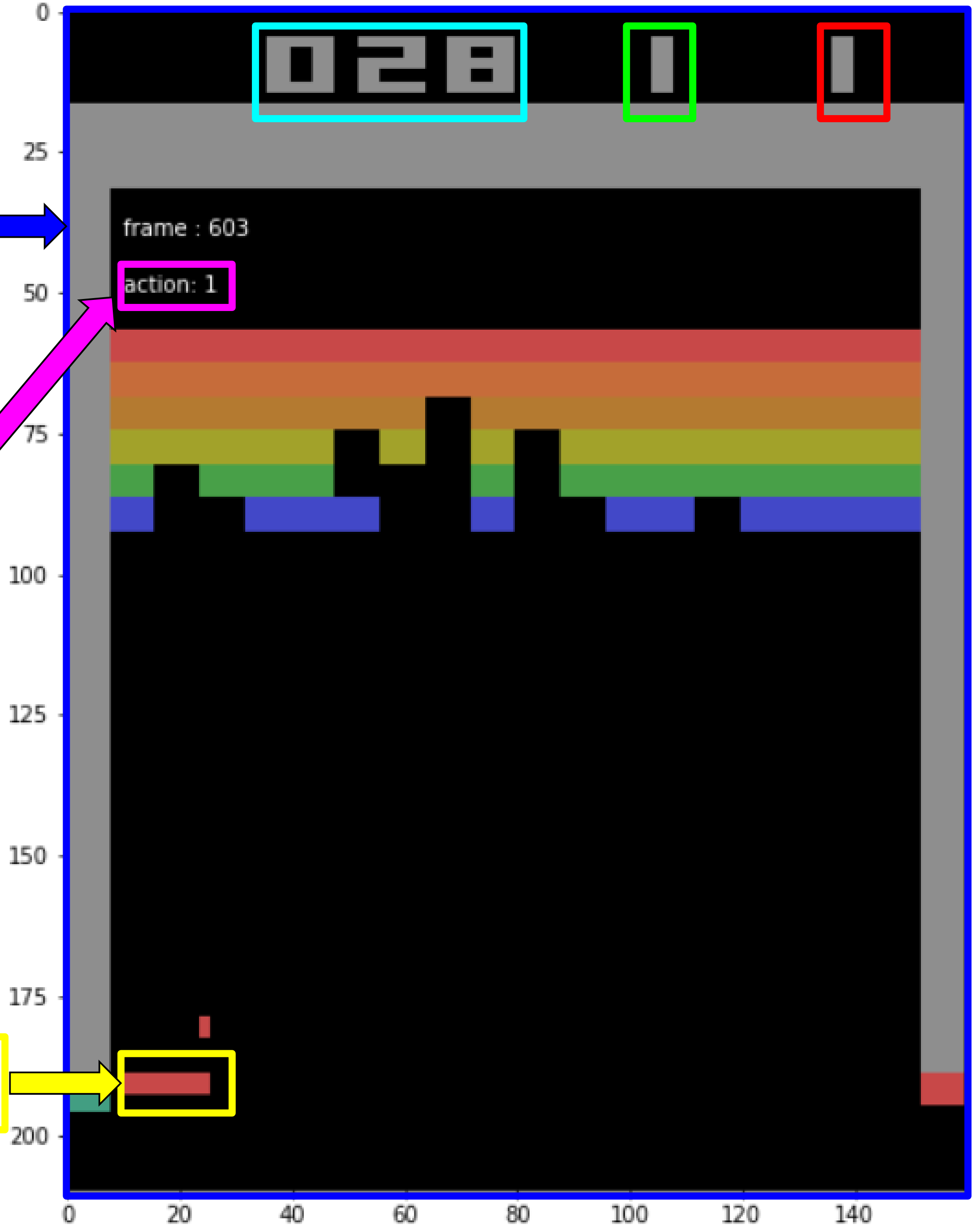
Training time: 15 hours including simulation time on a GTX 650 with 1 GB of RAM



Frame: snapshot of the environment state at every point

Action: a set of actions, that agent can take {0, 1, 2, 3}

Agent

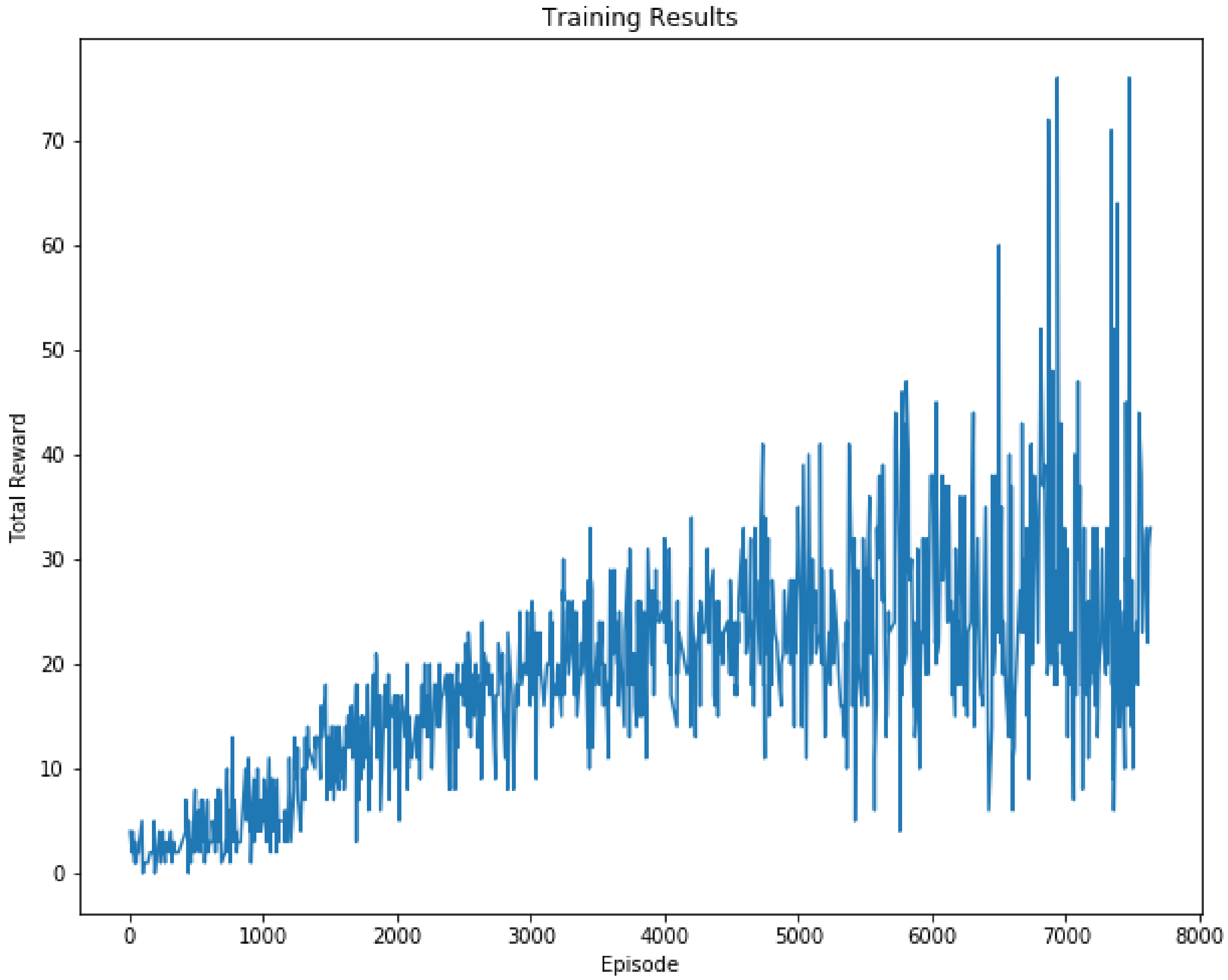


Score: evaluation metric

Number of “lives” for each game (initially 5)

Game’s level

DQN: Atari



Training Results
Total Reward
Episode
0
10
20
30
40
50
60
70
0
1000
2000
3000
4000
5000
6000
7000
8000

DQN: Tom and Jerry