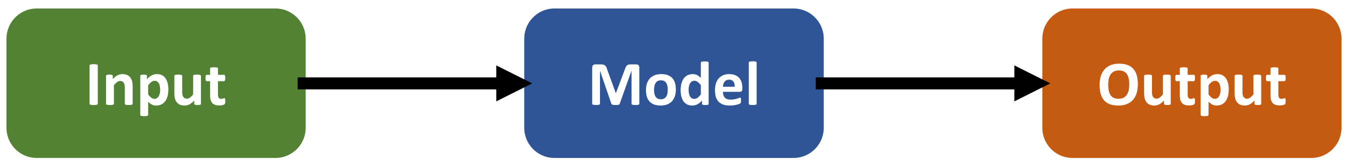
Hands-on Introduction to Deep Learning

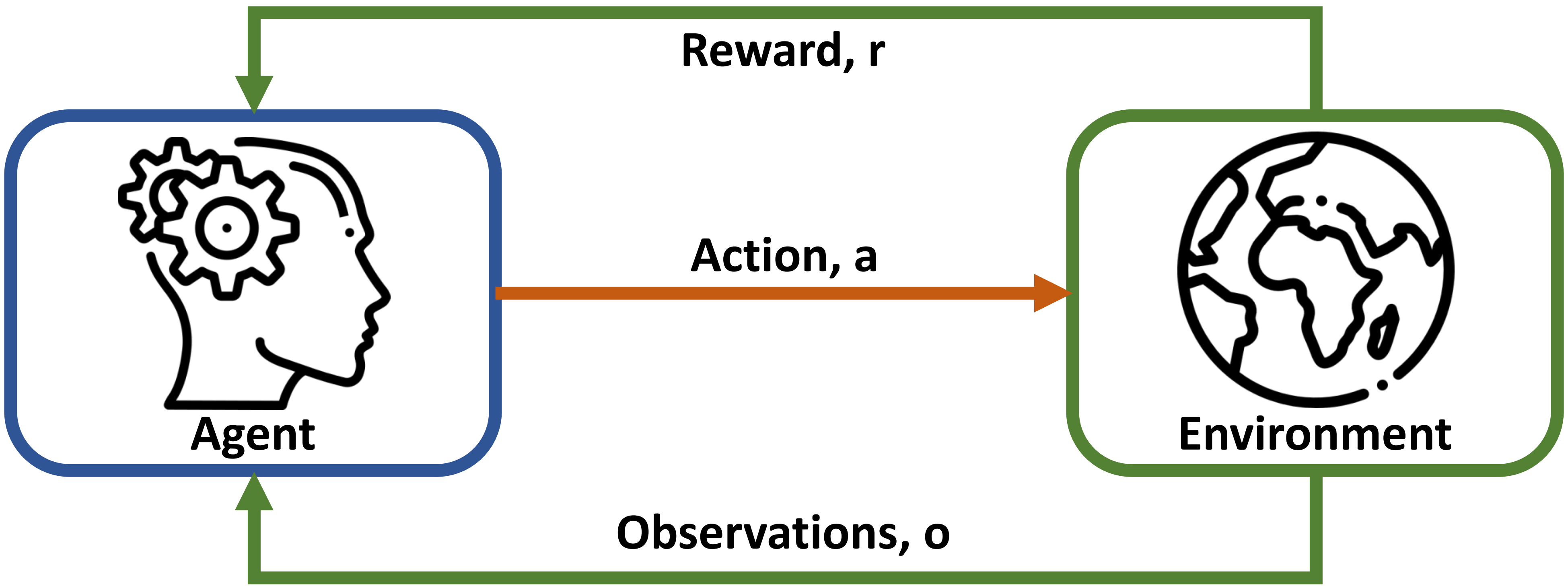
Deep Reinforcement Learning

cnrs

INSTITUT DU
DÉVELOPPEMENT ET DES
RESSOURCES EN
INFORMATIQUE
SCIENTIFIQUE

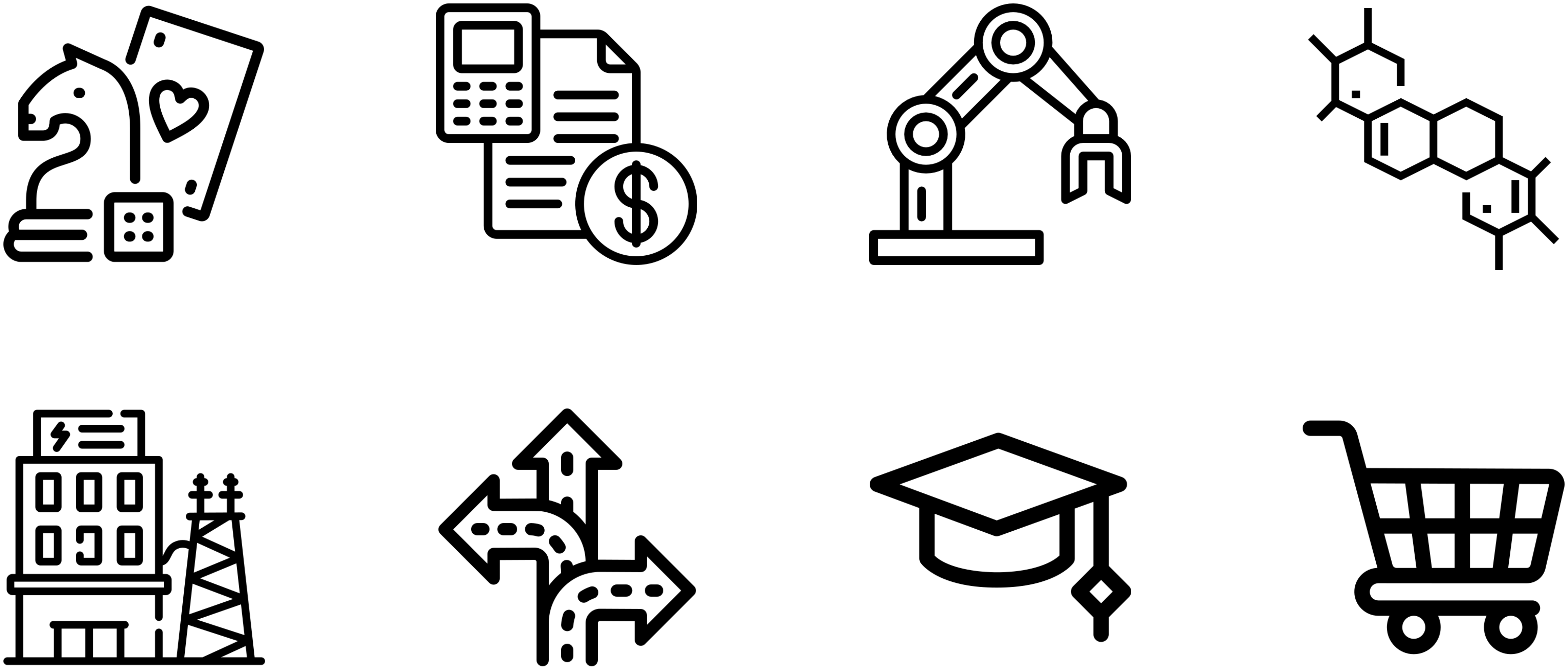


Input
Model
Output

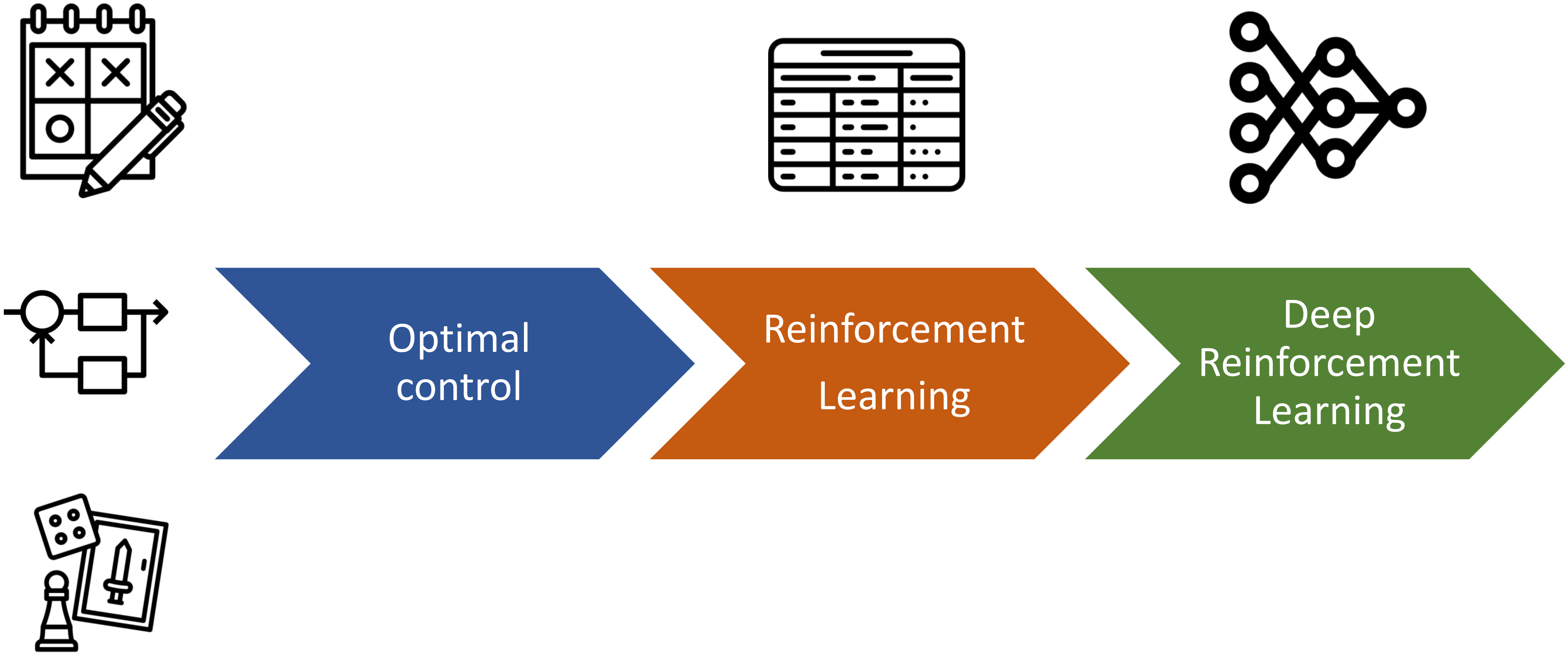


Reward, r
Agent
Action, a
Environment
Observations, o

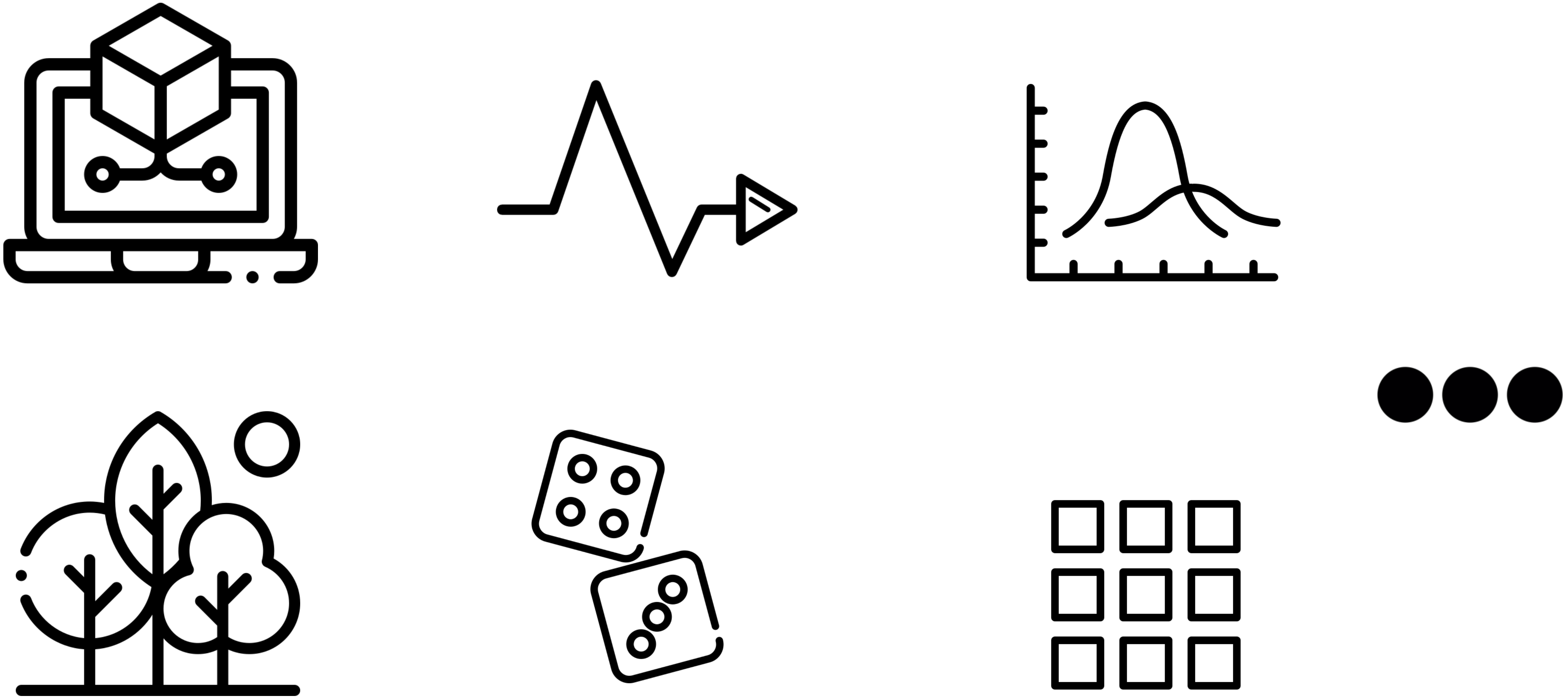
Context



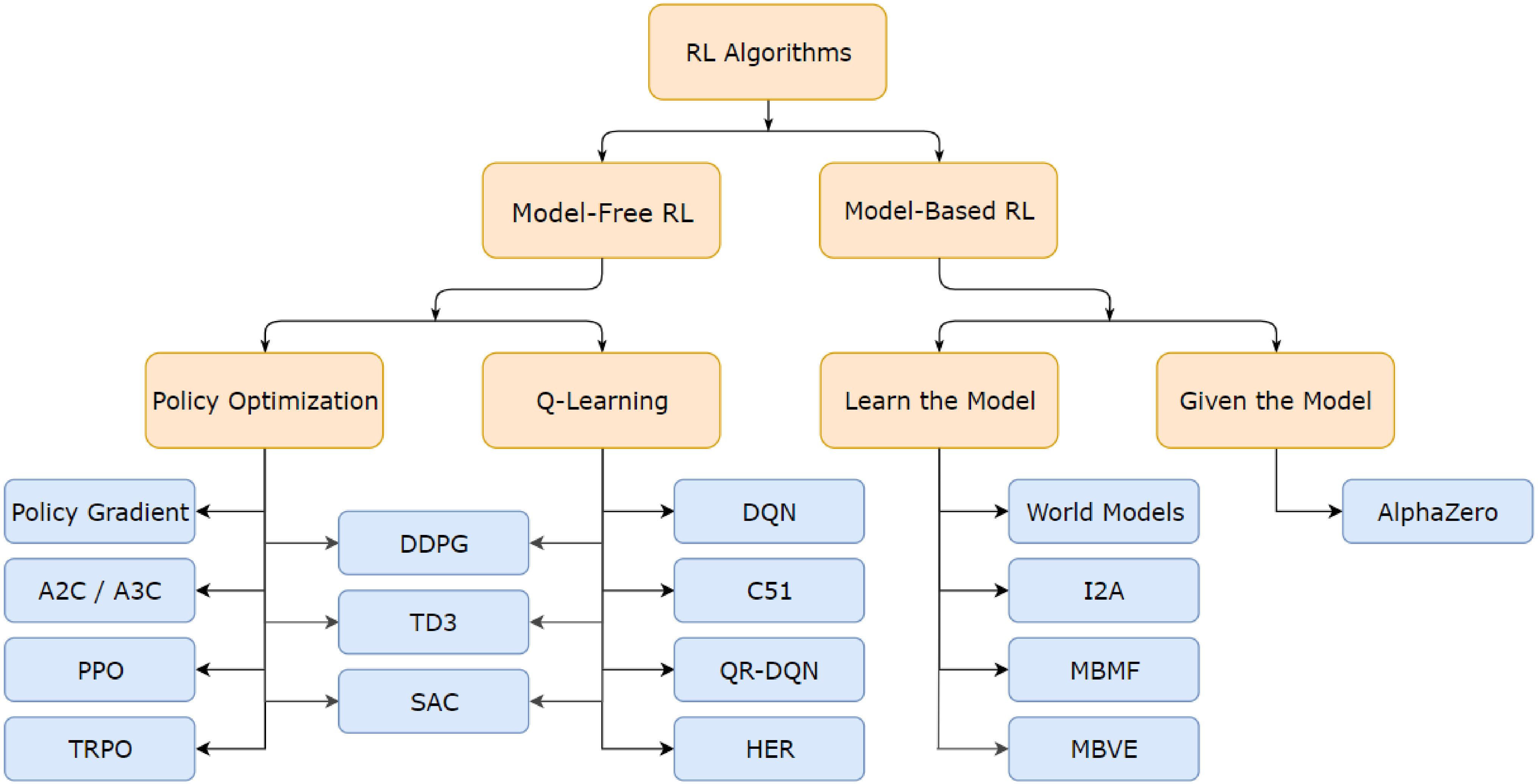
Applications



History of Reinforcement Learning

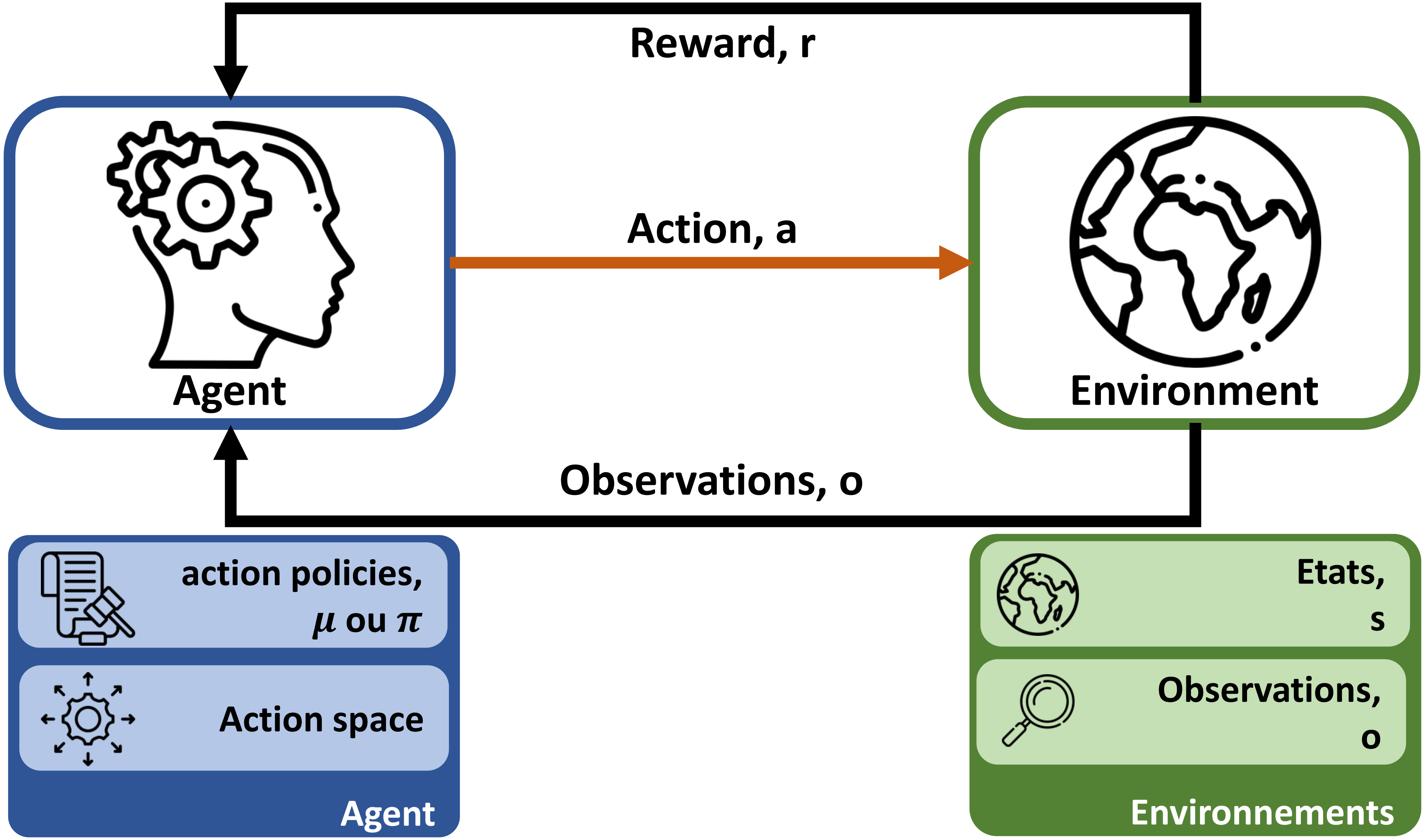


Environment



Part 2: Kinds of RL Algorithms — Spinning Up documentation. Spinningup.openai.com. (2022).

Taxonomy



Terminology

- **Trajectories:**

$$\tau = (s_0, a_0, s_1, a_1, ...)$$

- **Rewards:**

$$r_t = R(s_t, a_t, s_{t+1})$$

Finite-horizon undiscounted return

$$R(\tau) = \sum_{t=0}^{T} r_t$$

Infinite-horizon discounted return

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t$$

- **On-policy Value Function:**

$$V^{\pi}(s) = \underset{\tau \sim \pi}{\mathrm{E}} [R(\tau) \,|s_0 = s]$$

- **On-policy Action-Value (Q) Function:**

$$Q^{\pi}(s,a) = \underset{\tau \sim \pi}{\mathrm{E}} [R(\tau) \,|s_0 = s, a_0 = a]$$

- **Optimal :**

$$\max_{\pi}$$

- **Policies:**

$$a_t = \mu(s_t) \qquad a_t \sim \pi(\cdot|s_t)$$

Bellman Equations

$$V^{\pi}(s) = \underset{\substack{a \sim \pi \\ s' \sim P}}{\mathrm{E}} [r(s,a) + \gamma V^{\pi}(s')]$$

$$Q^{\pi}(s,a) = \underset{s' \sim P}{\mathrm{E}} \left[r(s,a) + \gamma \underset{a' \sim \pi}{\mathrm{E}} [Q^{\pi}(s',a')] \right]$$

Concepts

Monte-Carlo

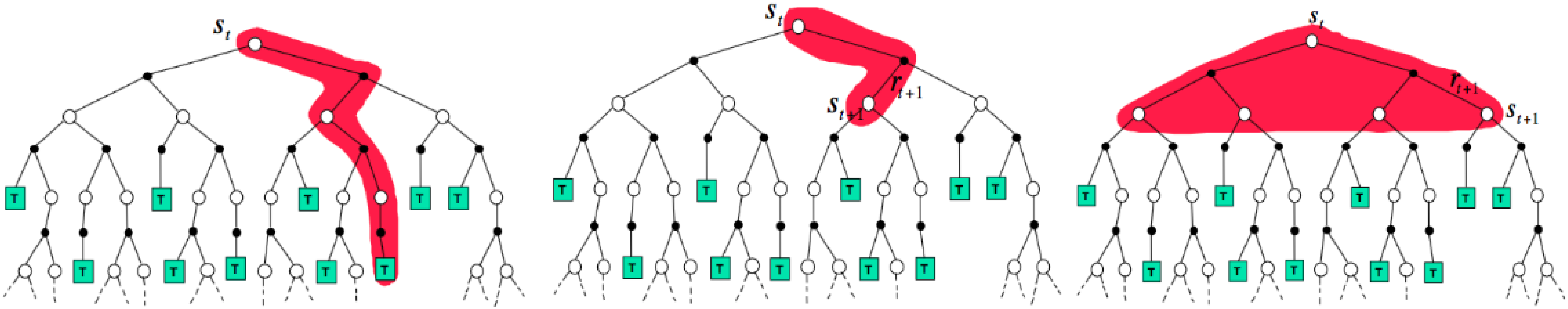
$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t - V(S_t)\right)$$

Temporal-Difference

$$V(S_t) \leftarrow V(S_t) + \alpha \left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)$$

Dynamic Programming

$$V(S_t) \leftarrow \mathbb{E}_\pi \left[R_{t+1} + \gamma V(S_{t+1})\right]$$



David Silver's RL course lecture 4: "Model-Free Prediction"

Monte Carlo | Temporal Difference | Dynamic Programming

On Policy:

- Same policy used to generate experiences and to improve
- **SARSA** $$Q(a,s) \leftarrow Q(a,s) + \alpha \cdot \left(r_s + \gamma \cdot Q(a',s') - Q(a,s)\right)$$

Off Policy:

- One policy (Target policy) to generate samples
- Another different policy optimized during the process
- **Q Learning** $$Q(a,s) \leftarrow Q(a,s) + \alpha \cdot \left(r_s + \gamma \max_{a'} Q(a',s') - Q(a,s)\right)$$

On | Off Policies

Set values for learning rate α、 discount rate γ、 reward matrix R

Initialize $Q(s,a)$ to zeros

Repeat for each episode,do

  Select state s randomly

  Repeat for each step of episode,do

    Choose a from s using ε-greedy policy or Boltzmann policy

    Take action a obtain reward r from R, and next state s'

    Update $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$

    Set $s = s'$

    Until s is the terminal state

  End do

End do

Q Table	Actions
Etats	

Q Learning

Policy parameters optimization:

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\pi_{\theta_k})$$

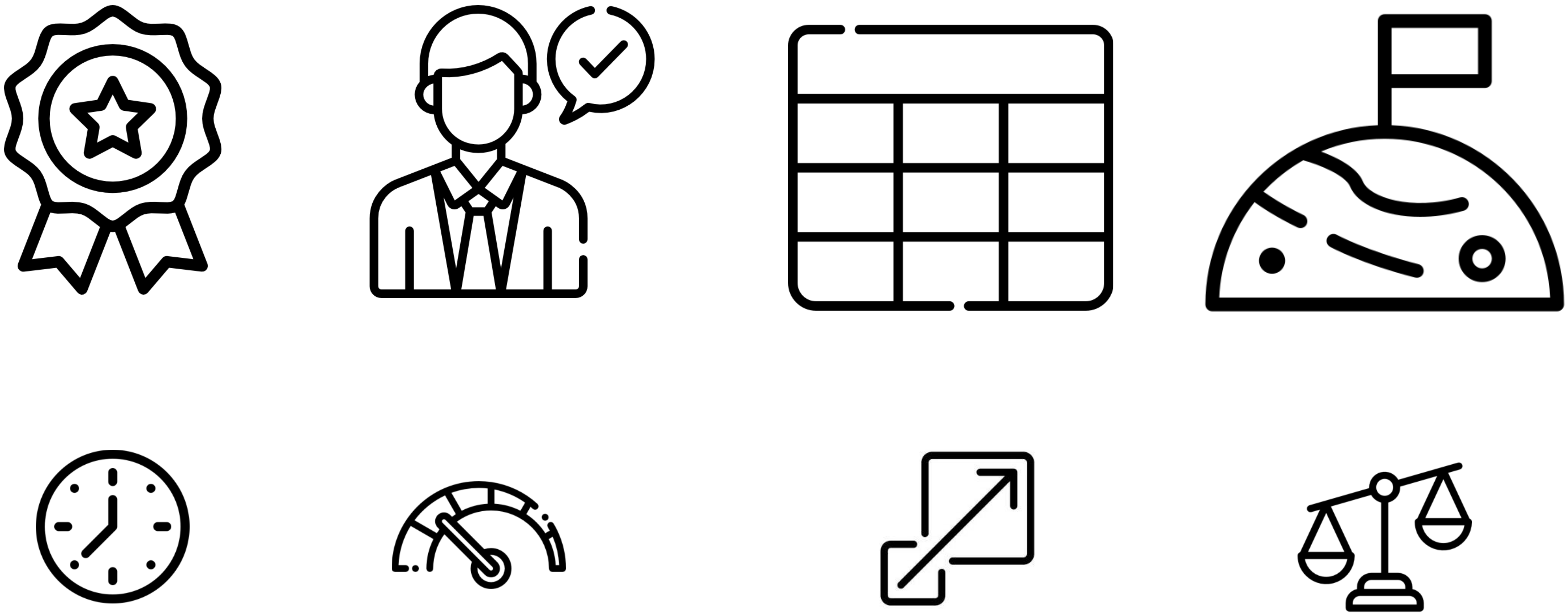
Gradient of expected finite-horizon:

$$\nabla_\theta J(\pi_\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) A^{\pi_\theta}(s_t, a_t)\right]$$

Advantage function:

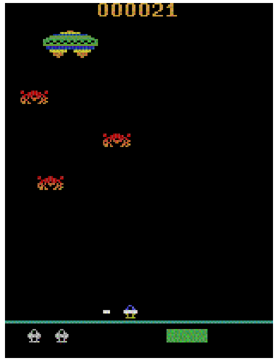
$$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$$

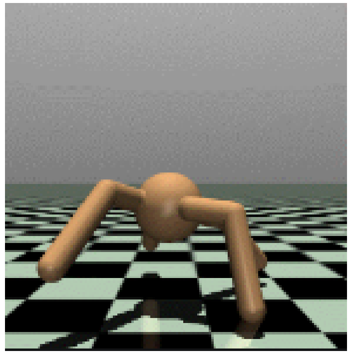
Policy Optimization – Vanilla Policy Gradients

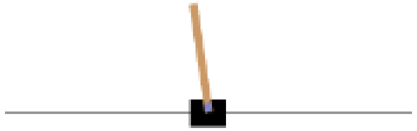


Reinforcement Learning

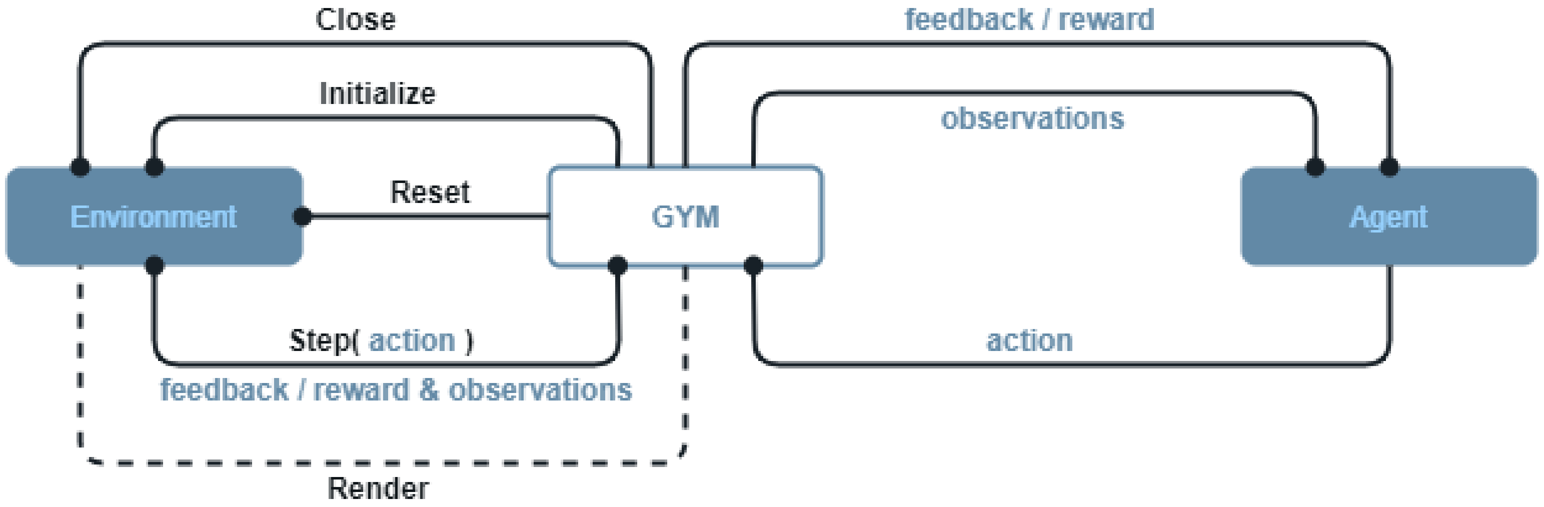
Gym



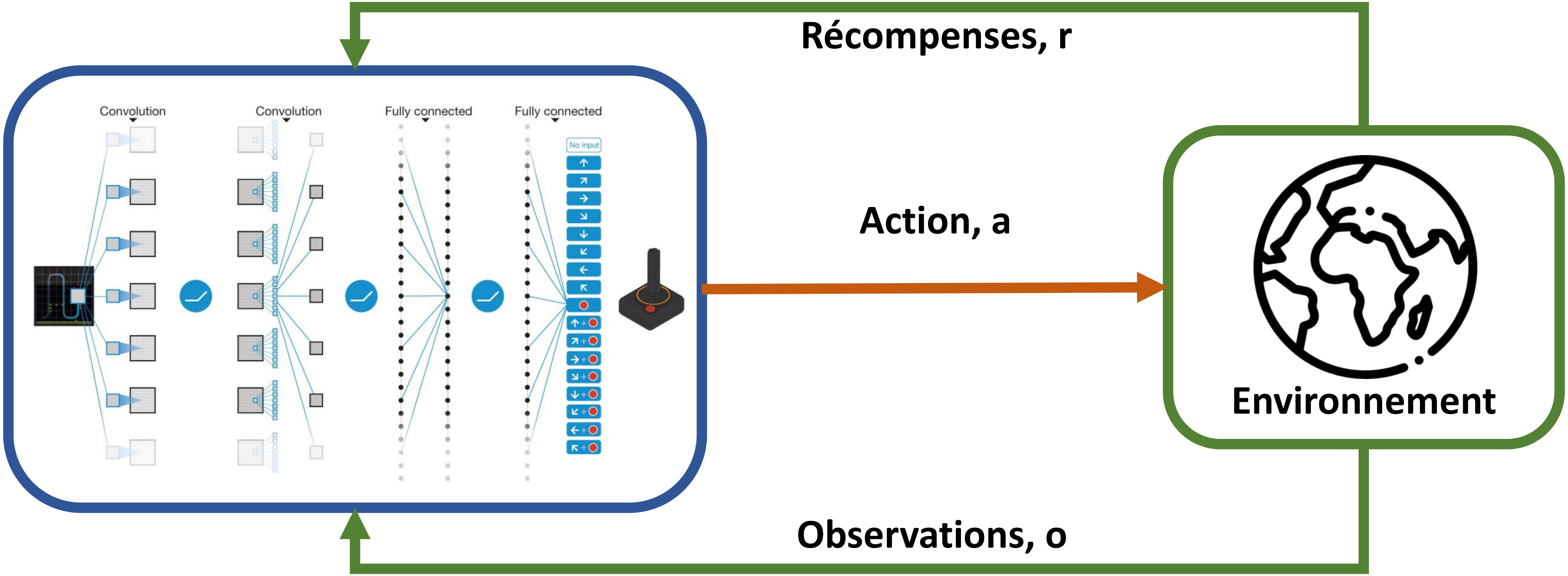




Atari
MuJoCo
Toy Text
Classic Control
Box2D
Third Party Environments

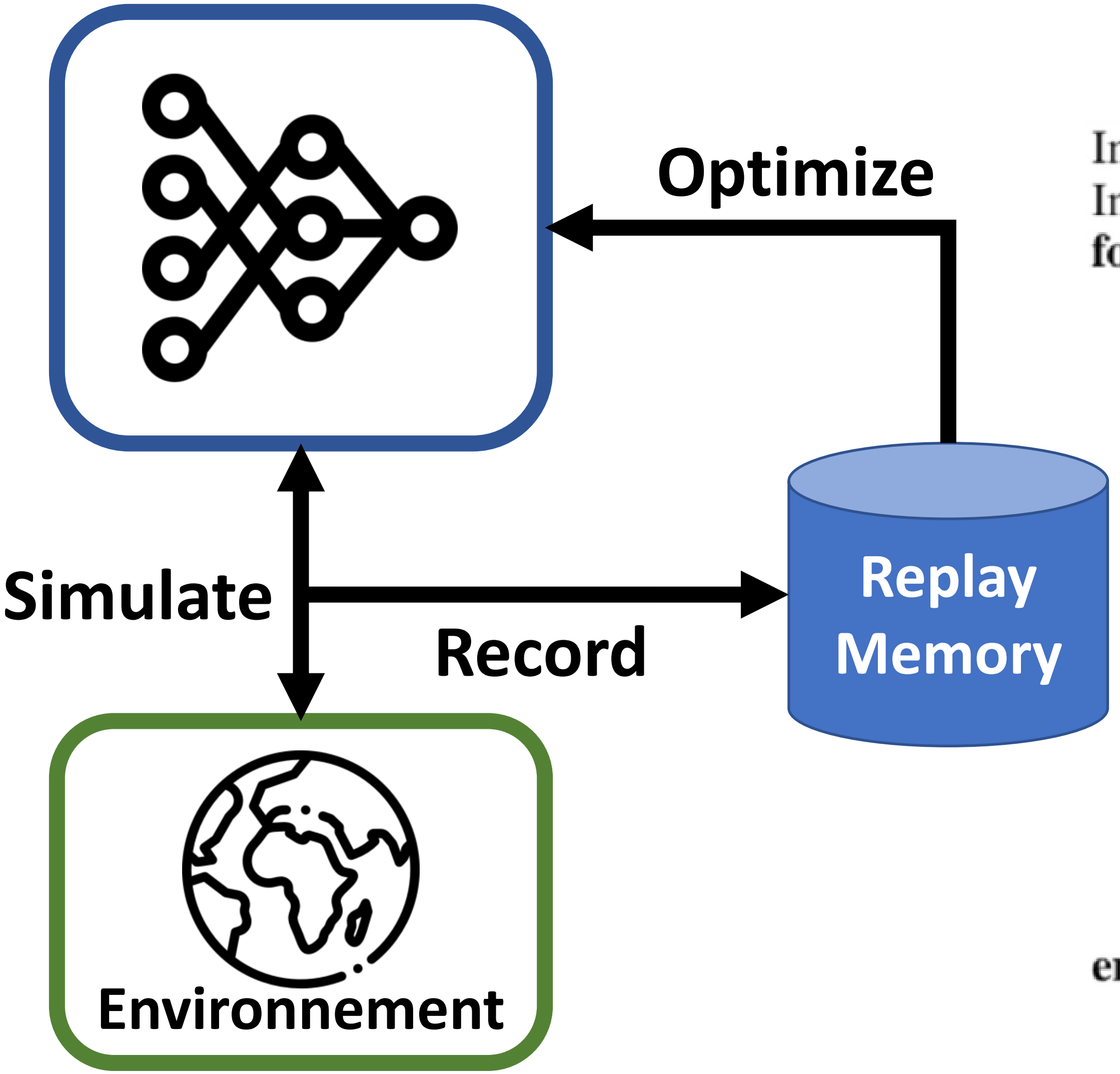


Simulated environments



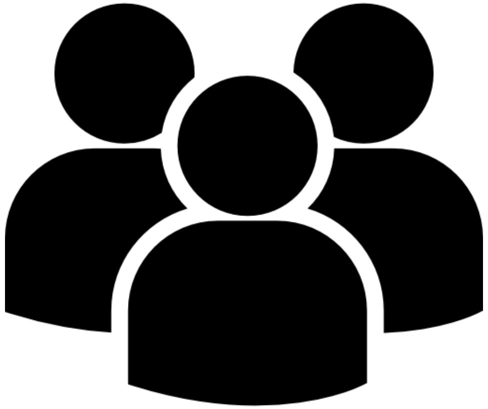
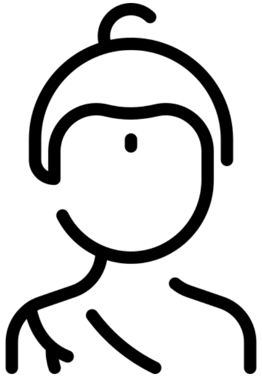
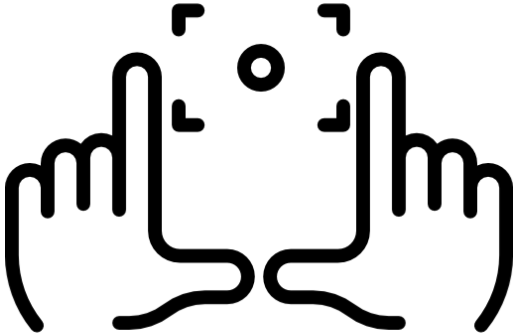
Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning."

Deep Q Learning

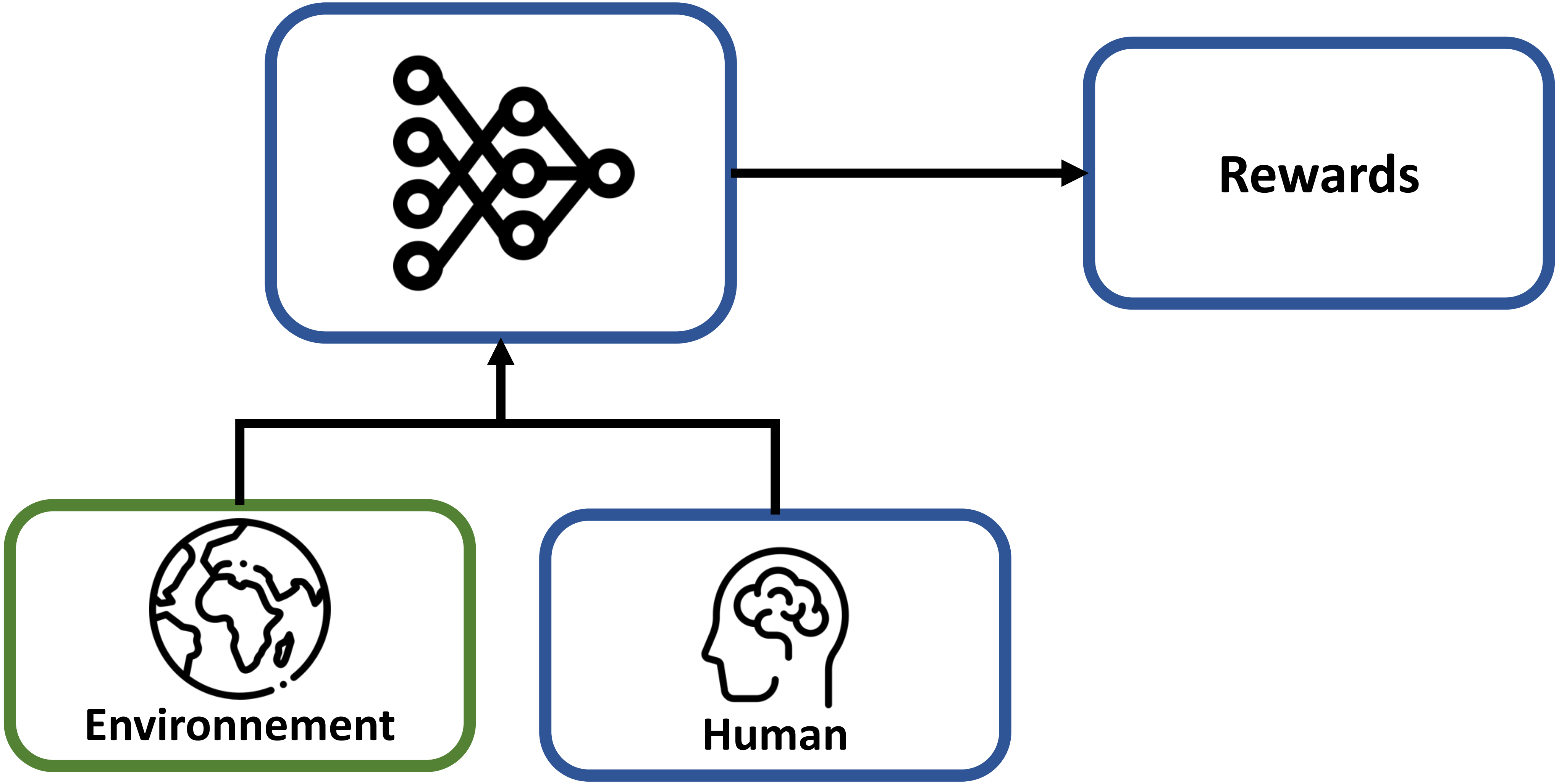


Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
    Initialise state s_t
    for $t = 1, T$ **do**
        With probability ϵ select a random action a_t
        otherwise select $a_t = \max_a Q^*(s_t, a; \theta)$
        Execute action a_t and observe reward r_t and state s_{t+1}
        Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
        Set $s_{t+1} = s_t$
        Sample random minibatch of transitions (s_t, a_t, r_t, s_{t+1}) from $\mathcal{D}$
        Set $y_j = \begin{cases} r_j & \text{for terminal } s_{t+1} \\ r_j + \gamma \max_{a'} Q(s_{t+1}, a'; \theta) & \text{for non-terminal } s_{t+1} \end{cases}$
        Perform a gradient descent step on $(y_j - Q(s_t, a_j; \theta))^2$
    end for
end for

Deep Q Learning - Training

Deep Reinforcement Learning



Inverse Reinforcement Learning