



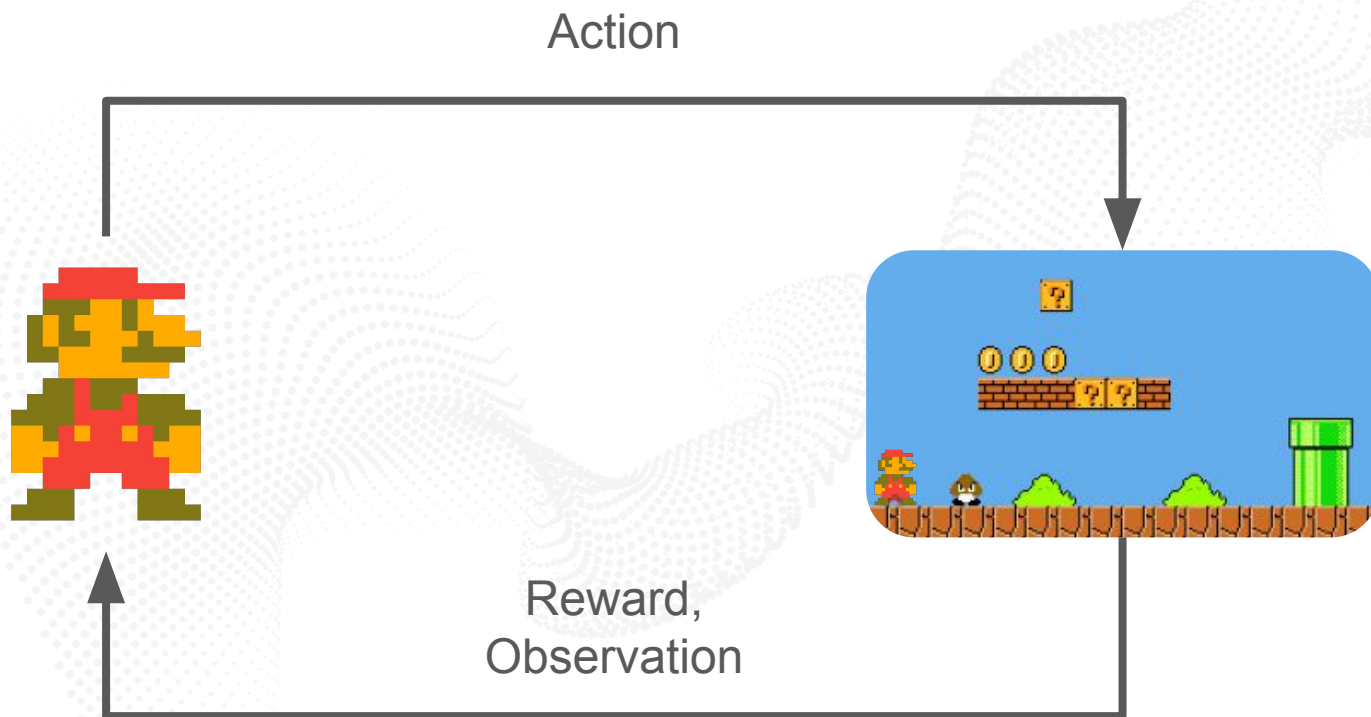
Hands-On RL: Creating 2048 for RL Agent

World Learning Algeria
24 April, 2025

Omayma Mahjoub - Research Engineer, InstaDeep

The RL Framework

The RL Framework



Markov Decision Process (MDP)

Classical formulation of sequential decision-making systems.

A (finite) MDP consists of a tuple $(\mathbf{S}, \mathbf{A}, \mathbf{P}, \mathbf{R}, \rho, \gamma)$ where:

- \mathbf{S} 🧱: a (finite) set of states
- \mathbf{A} 🎮: a (finite) set of possible actions
- \mathbf{P} 🔄: $\mathbf{S} \times \mathbf{A} \times \mathbf{S} \rightarrow \mathbb{R}$ is the transition probability distribution
- \mathbf{R} 💰: $\mathbf{S} \times \mathbf{A} \rightarrow \mathbb{R}$ is the reward function with $r_t = R(s_t, a_t)$
- ρ 🎲: $\mathbf{S} \rightarrow \mathbb{R}$ is the distribution of the initial state s_0
- γ ⌚: the discount factor

📖 **Markov Property:** The probability of future states depends only on the present state—not the past.

Markov Property

Markov Property: The probability distribution of the future states conditioned on the present and past values depends only upon the present state.

What happens next depends only on where you are now — not on how you got there!

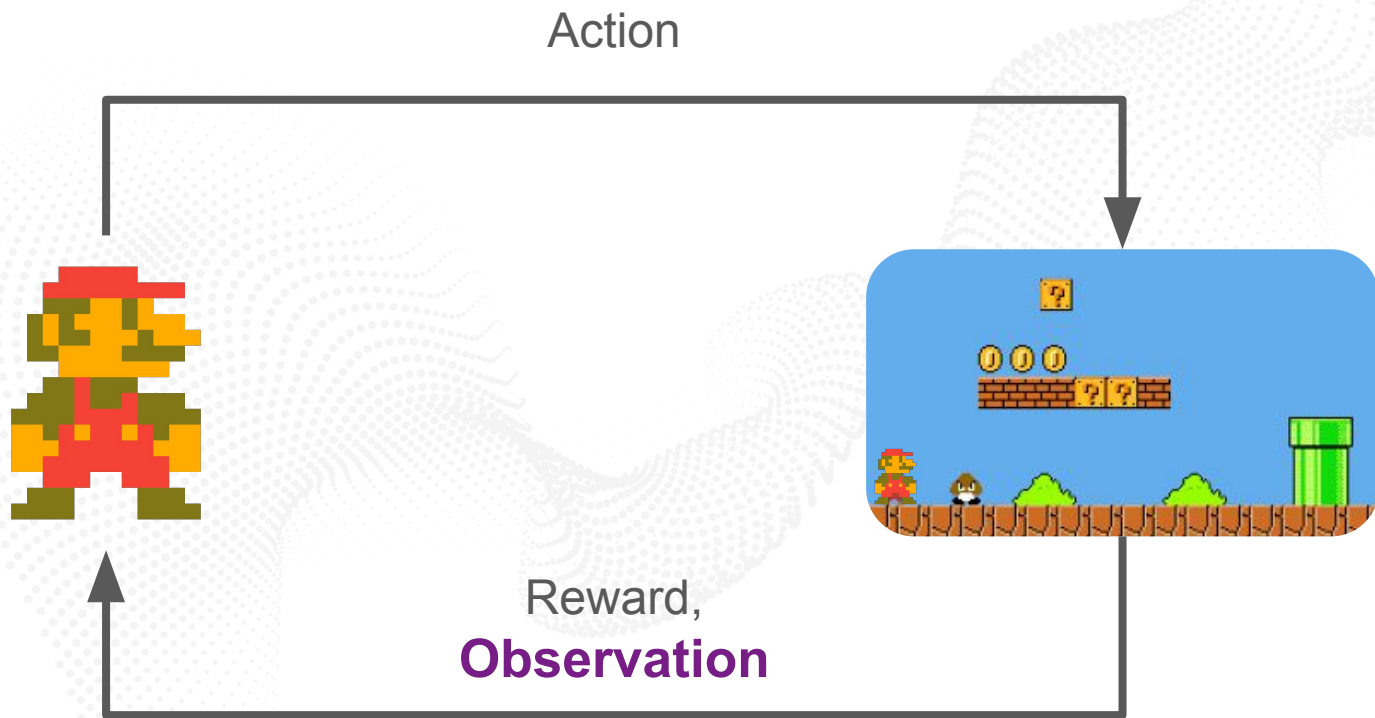
Markov Property for MDPs:

$$\begin{aligned} P(\text{Next State} \mid \text{Last State, Last Action, Two states ago, ...}) = \\ P(S_{t+1} \mid S_t, A_t, S_{t-1}, A_{t-1}, \dots, S_0) = \\ P(S_{t+1} \mid S_t, A_t) \end{aligned}$$



Andrey "GOATee"
Markov
(1856-1922)

States/Observations



States/Observations

Observations/States are the information our agent gets from the environment.

- **State:** is a **complete description** of the state of the world.
- **Observation:** is a **partial description** of the state. In a partially observed environment.



→The chess game is a **fully observable** environment.

The agent receive a **state** from the environment since it has access to the whole check board information.

Observation = State

States/Observations

Observations/States are the information our agent gets from the environment.

- **State:** is a **complete description** of the state of the world.
- **Observation:** is a **partial description** of the state. In a partially observed environment.

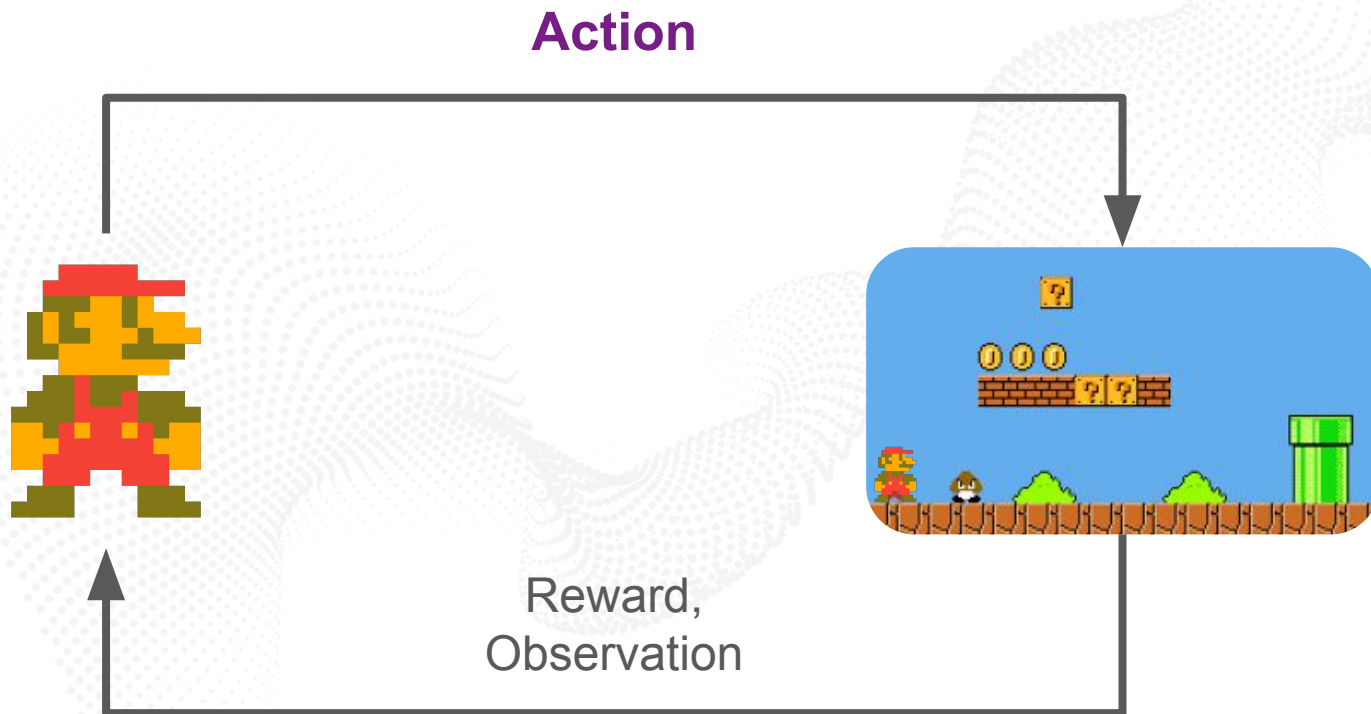


→ Mario game is a **partial observable** environment.

The agent can only view its the current frame and does not have visibility of the full journey in that level, so it receives **an observation**.

Observation \subset State

Actions



Actions

The Action space is the set of all possible actions in an environment.

- **Discrete space:** the number of possible actions is finite.
- **Continuous space:** the number of possible actions is infinite.
- **Hybrid actions:** mix of both.



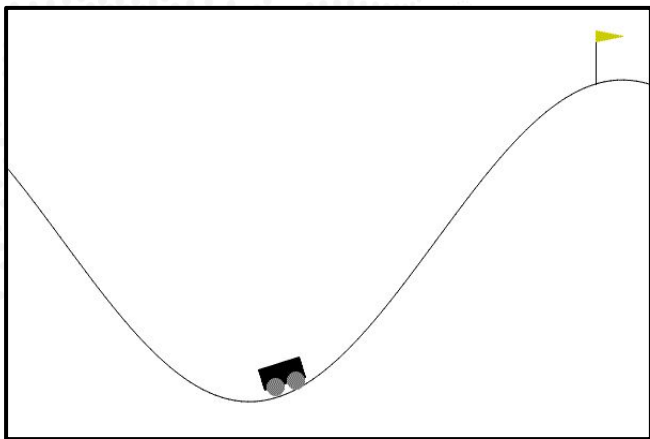
Discrete actions

Source: Hugging Face
Reinforcement
Learning course ([link](#))

Actions

The Action space is the set of all possible actions in an environment.

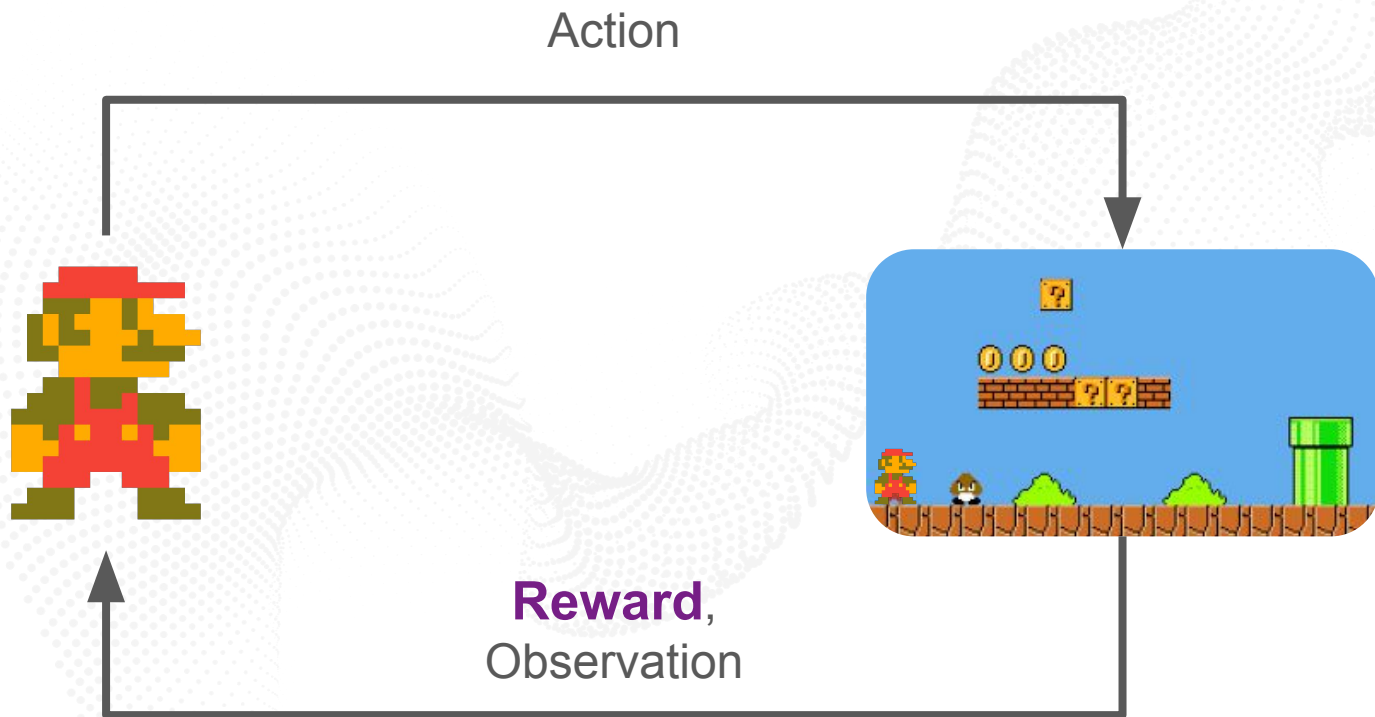
- **Discrete space:** the number of possible actions is finite.
- **Continuous space:** the number of possible actions is infinite.
- **Hybrid actions:** mix of both.



The action is in range $[-1,1]$ representing the directional force applied on the car.

Continuous actions

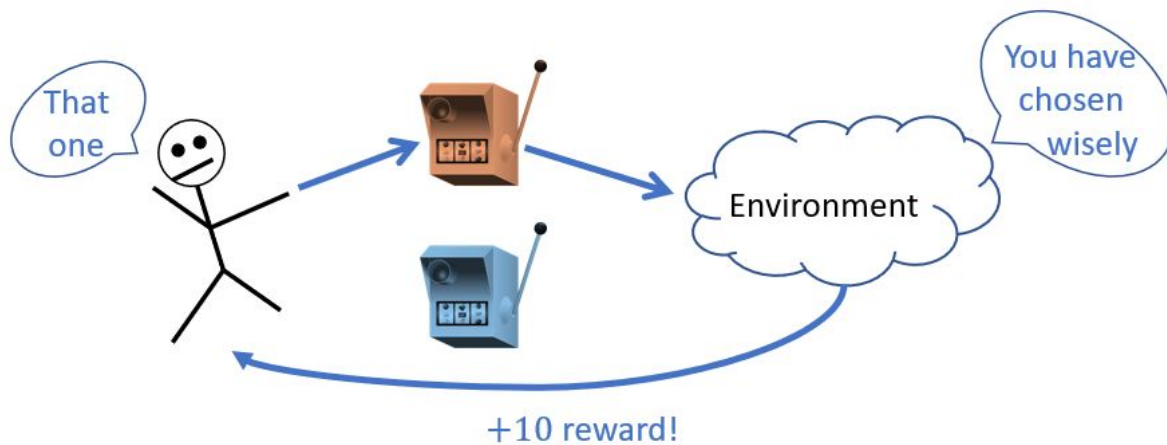
Reward



Rewards

The reward is a scalar value we obtain periodically from the environment: $r_t = R(s_t, a_t)$

- **Frequency:** it can be every second (dense reward) or once in a lifetime (sparse reward).



Source: Andrew Forney, RL class [\(link\)](#)

RL Environment Components

Formulate the Problem as an MDP!

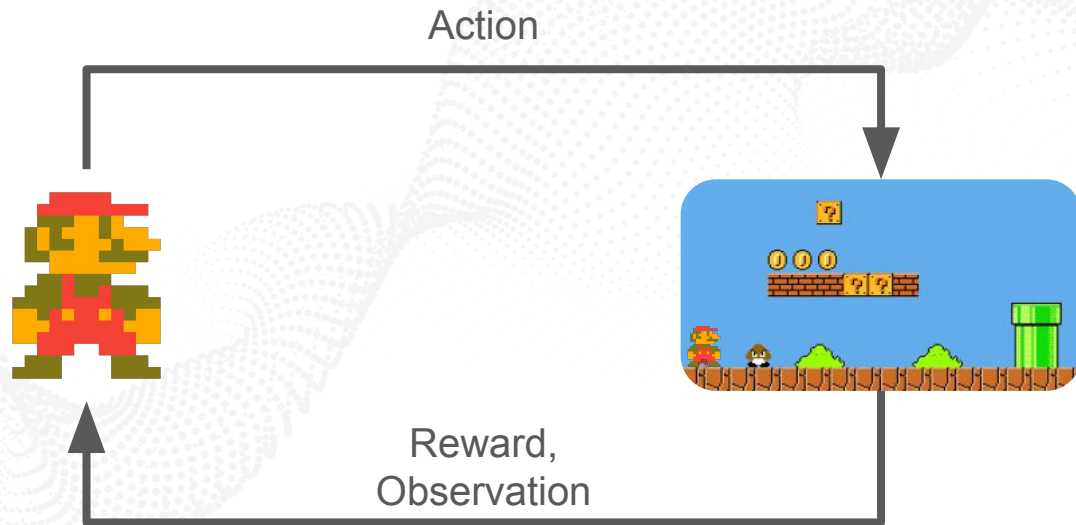
Goal: Finish the level.

State: Mario's position, enemies, obstacles, and items.

Observation: The current frame.

Action: Move, jump, or use items.

Reward: 1 for completing level, 0 otherwise.



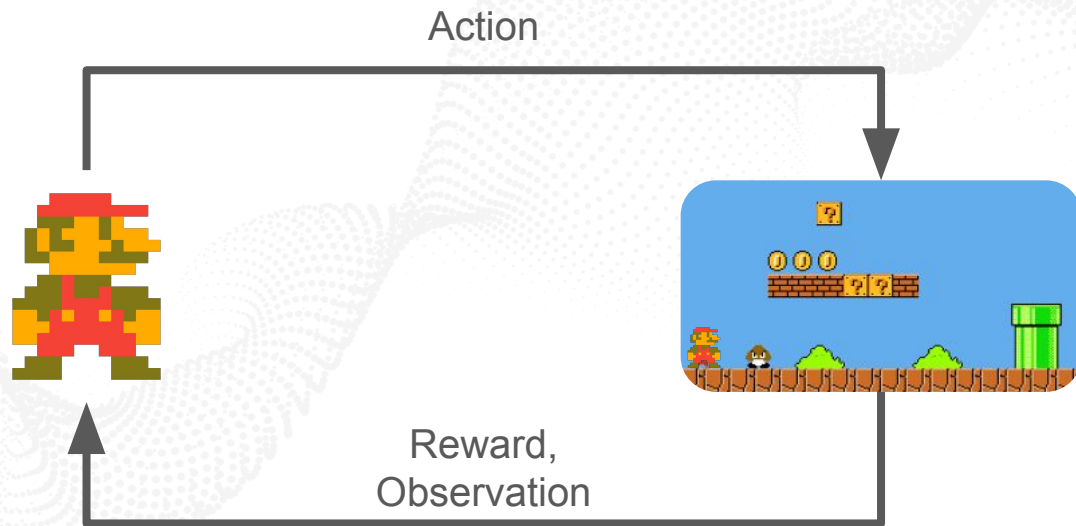
More Details!

Termination conditions:

- Reaching the end of the level
- Died because of enemies.
- Died because of falling.




Environment Dynamic:

- Enemies and obstacles positions per level.
- The movements of “Goomba” the mushroom...

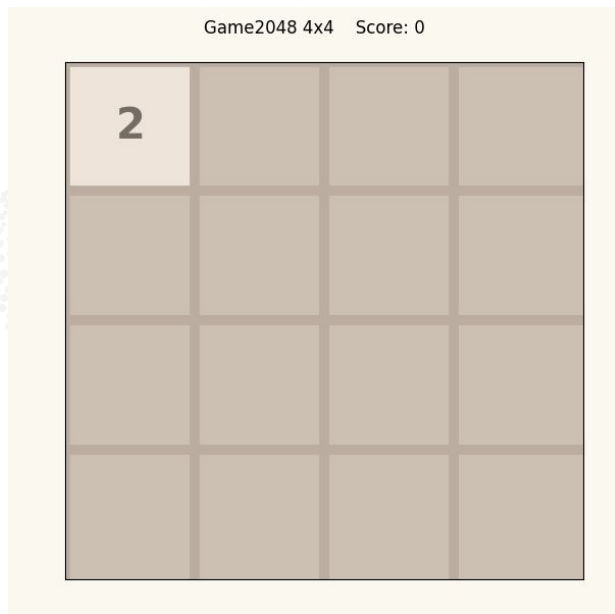


Environment Main Components

```
class ExampleEnv(gym.Env):  
    def __init__(self, **kwargs):  
        pass  
    def reset(self):  
        pass  
    def step(self, action):  
        pass  
    def render(self):  
        pass
```

- **reset()** 
Initializes the environment to a starting state.
Returns the initial observation.
- **step(action)** 
Takes an action and moves the environment forward.
Returns:
→ **next_state**, **reward**, **done**, **info**
- **render()** 
Visualizes the environment (optional).
Useful for debugging or demos.

Building the Game 2048



- 2048 is a popular single-player puzzle game that involves sliding numbered tiles on a 4x4 grid.
- The game starts with one random tiles (either 2 or 4) placed on the grid.
- After each turn, a new tile is added to the grid in a randomly chosen empty cell, with a value of either 2 or 4.
- The objective of the game is to combine tiles with the same number to create larger tiles, with the goal of reaching the 2048 tile, and ultimate one of reaching above 2048 tile.

<https://play2048.co/>

Colab Notebook: bit.ly/42Kyls6