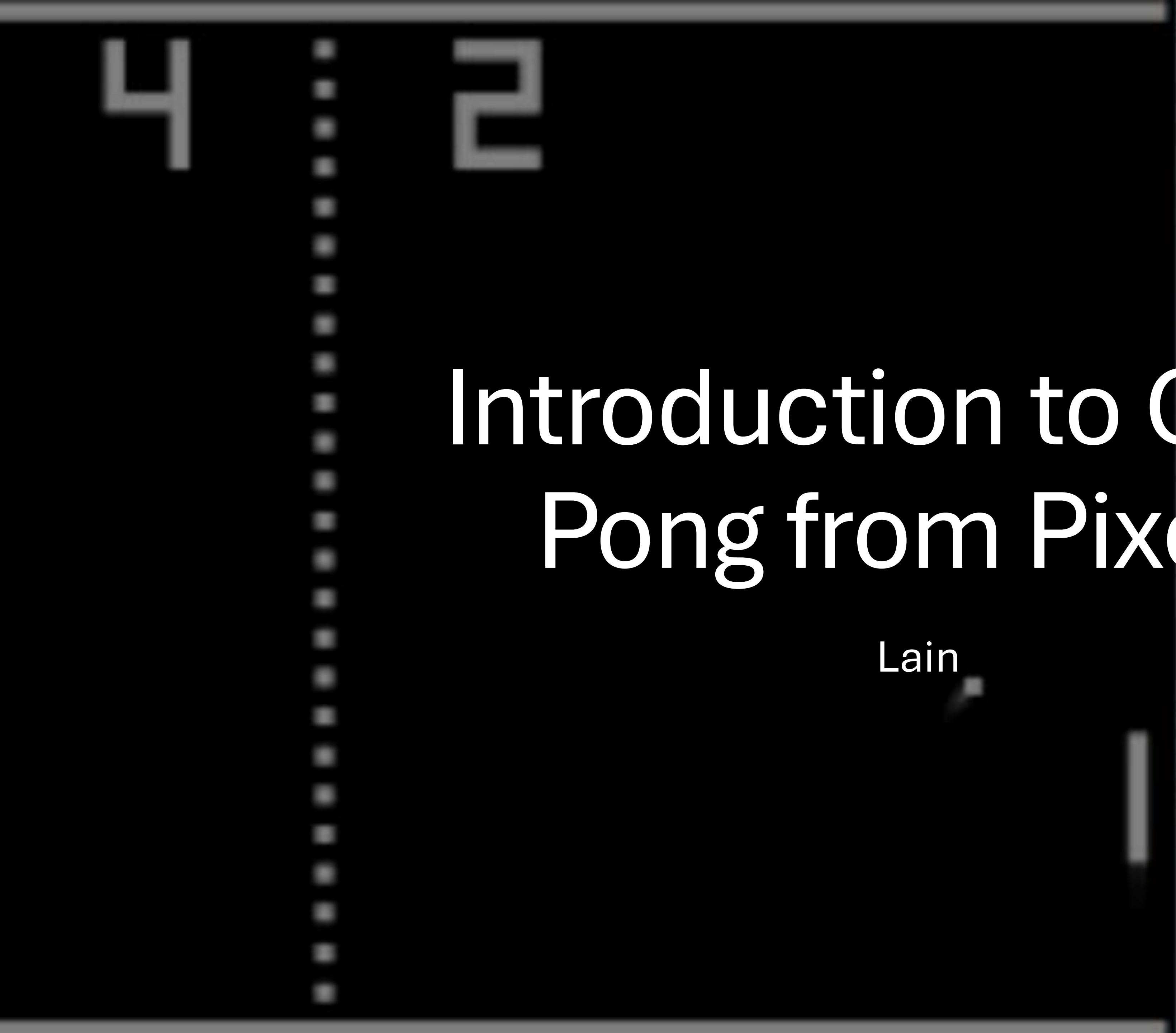
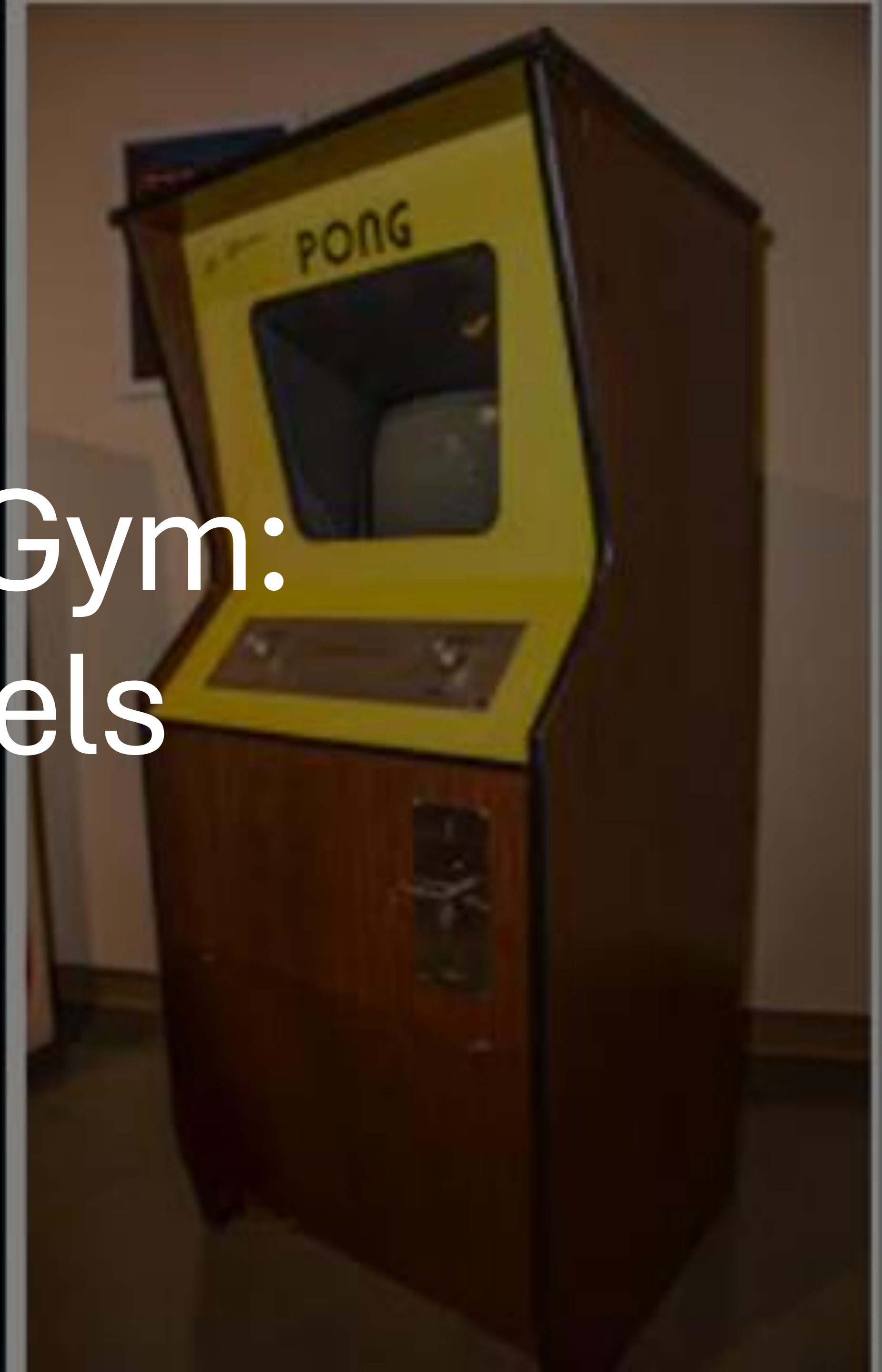


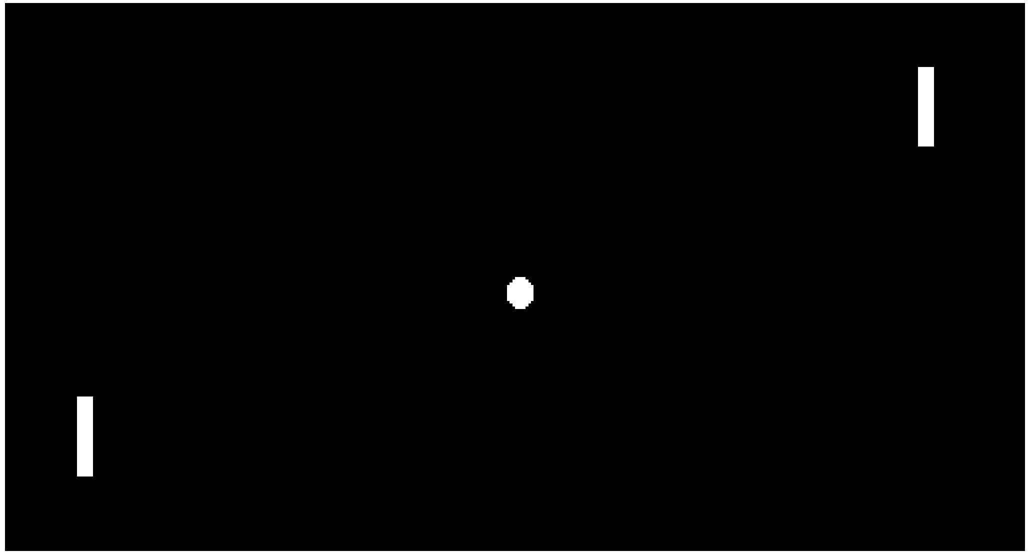



# Introduction to Gym: Pong from Pixels

Lain

# Overview

- Why Atari?
- Solving Pong without ML
- Solving Pong with RL
  - Markov Decision Processes (MDP)
  - Policies and Policy Networks
  - MDPs in Pong
- Gym/Gymnasium
- Demo



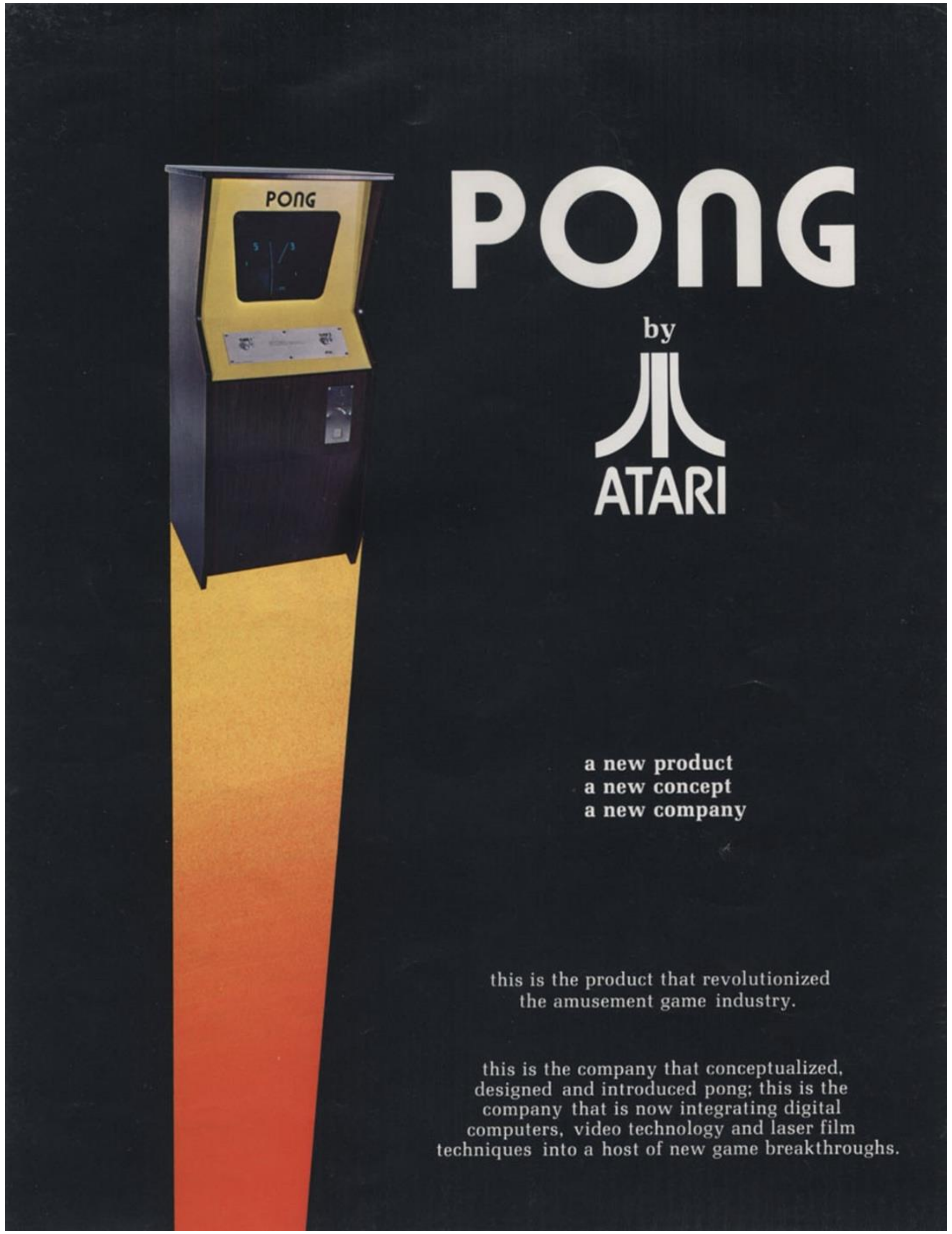
# Why Atari?

- **Origins:** The *Arcade Learning Environment (ALE)* bridging classic games and AI research

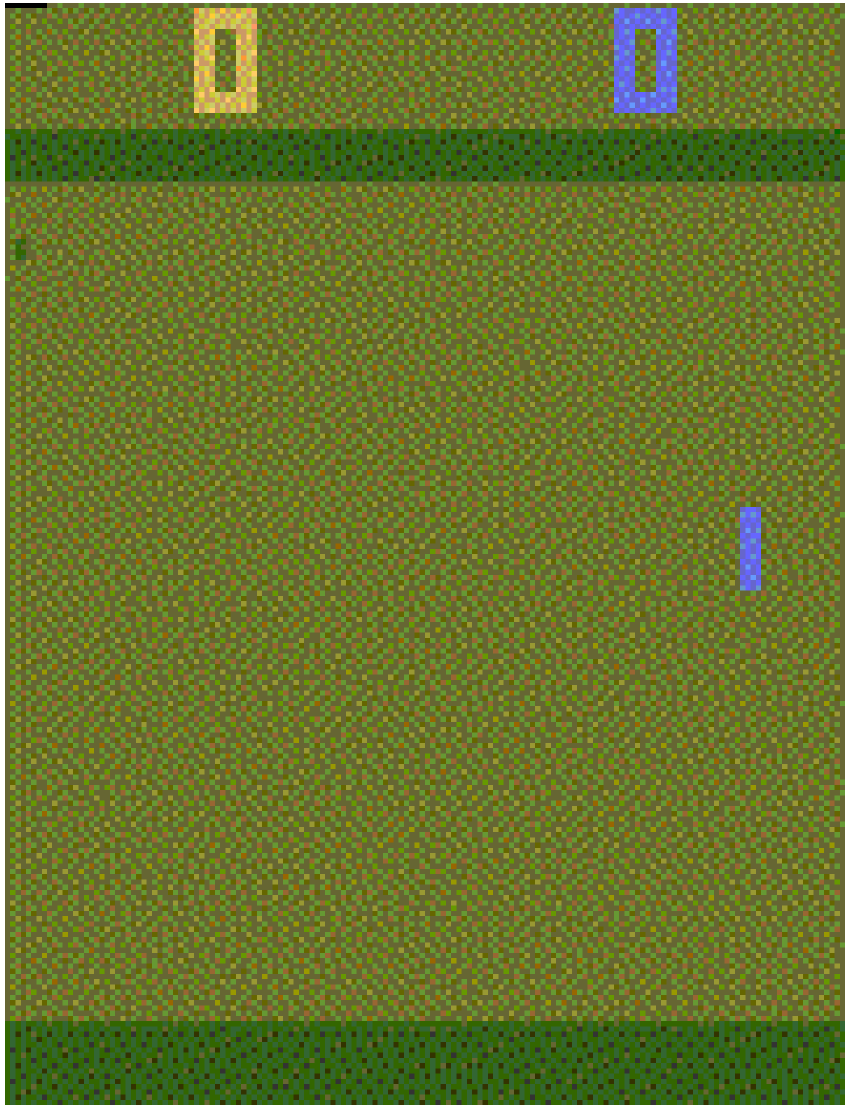
- **Why Atari for RL?**
  - Diverse tasks from high-dimensional pixel inputs
  - Challenging credit assignment & representation learning
  - Standardized evaluation & human performance baselines

- **Historical paper:** *Deep Q-Network (DQN)* achieving human level (Mnih et al. NeurIPS 2013; Nature 2015)

- Atari 2600 console (8-bit, 1977)
  - Fun tidbit: Wozniak worked on Pong; Wozniak and Jobs worked on Breakout together

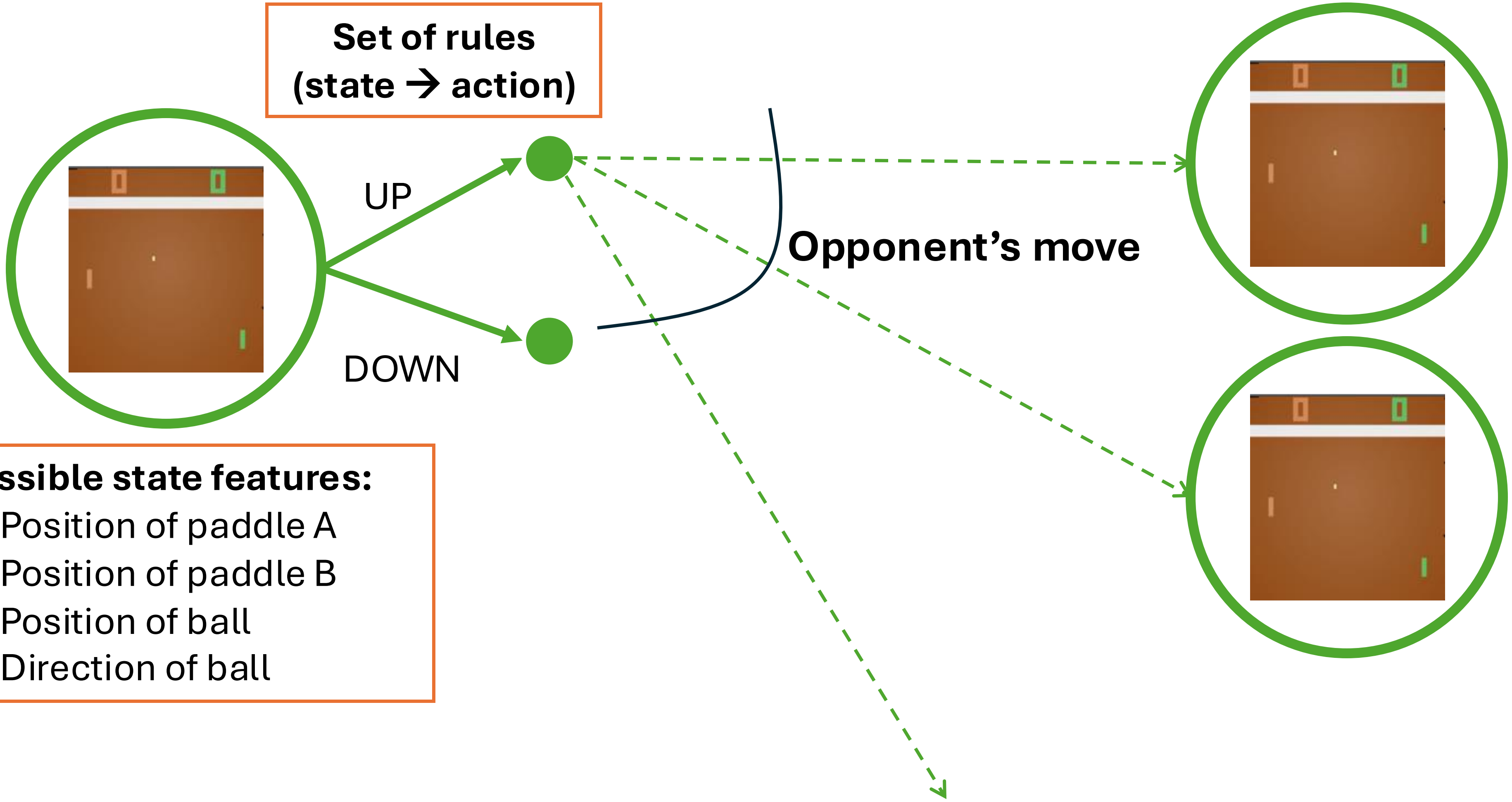


# Pong



https://gymnasium.farama.org/v0.26.3/environments/atari/pong

# Solving Pong without ML/RL

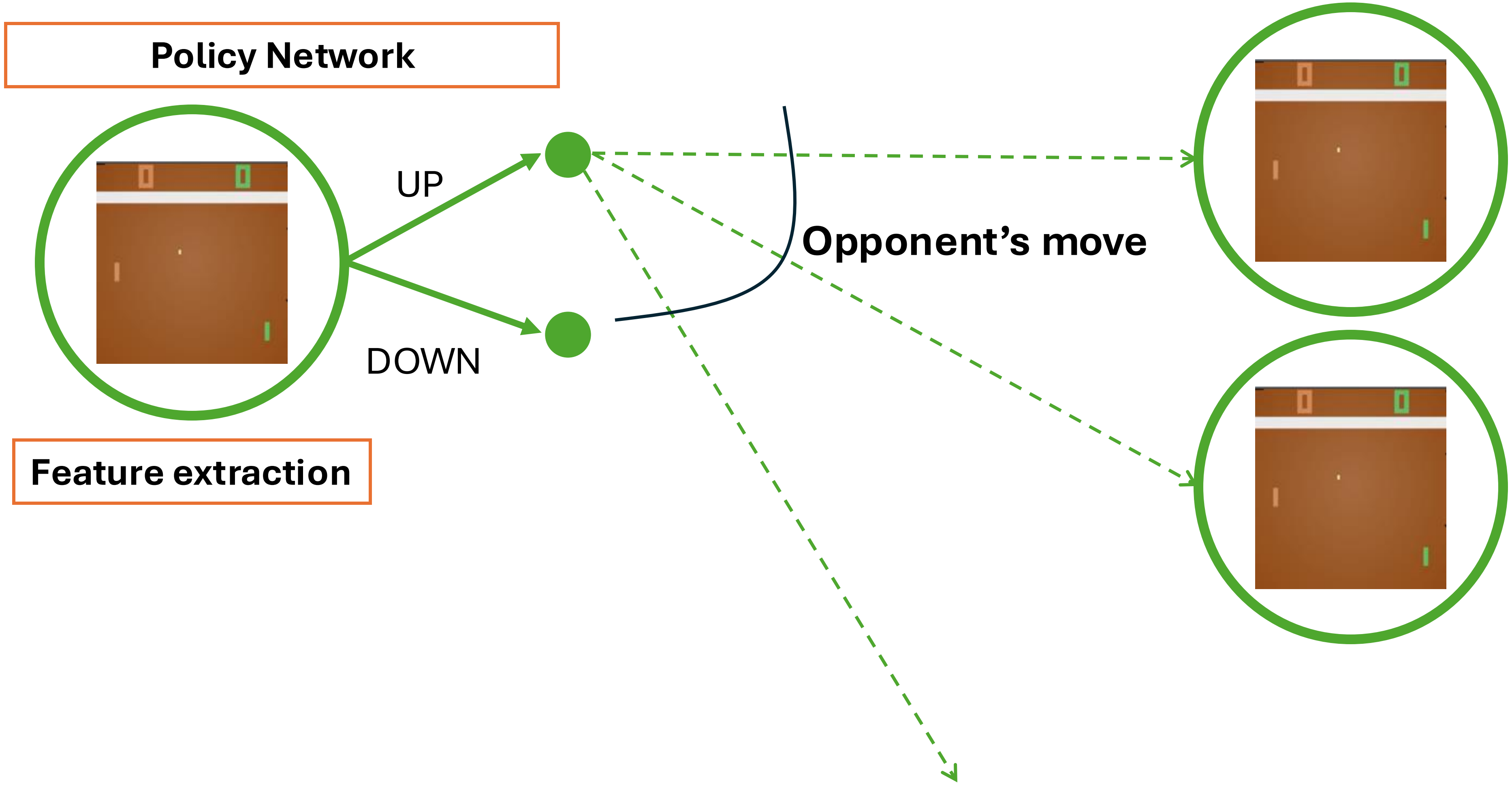



**Possible state features:**

- Position of paddle A
- Position of paddle B
- Position of ball
- Direction of ball

# Solving Pong with ML/RL

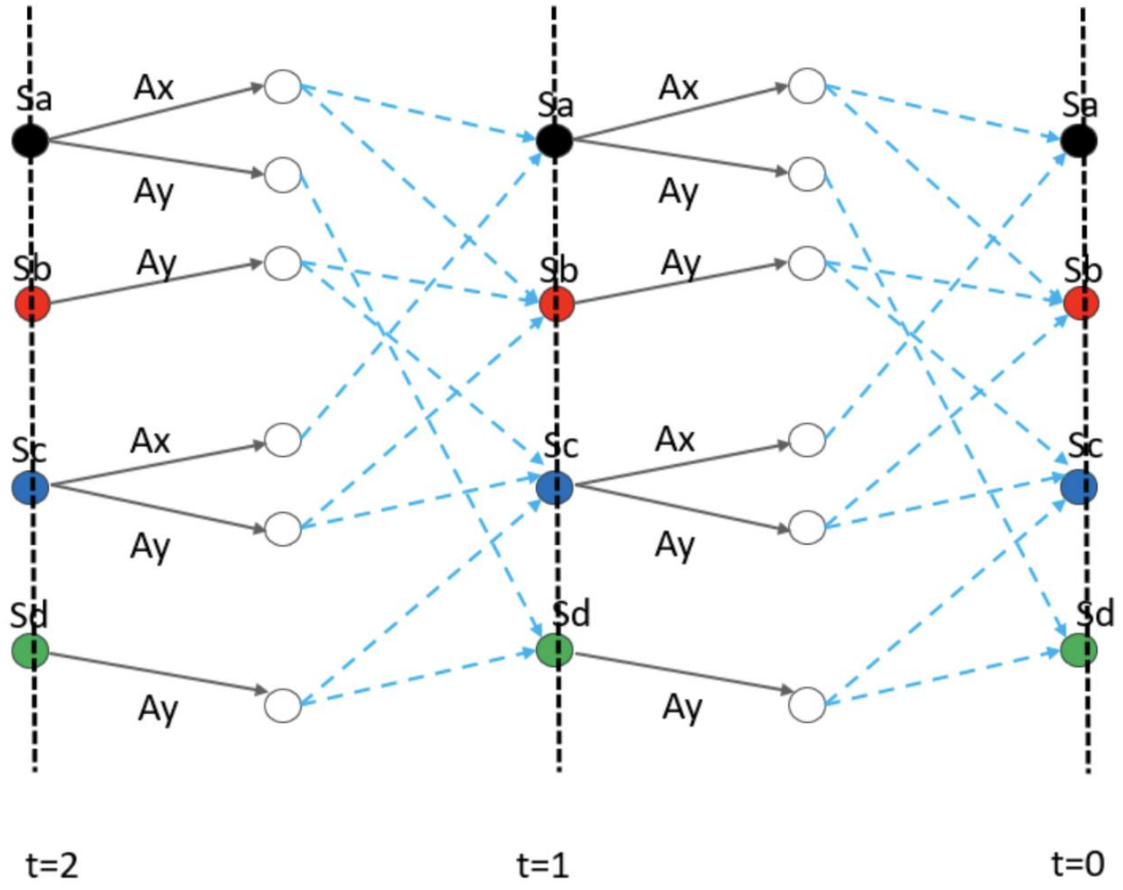
We formulate the problem using Markov Decision Processes (MDPs):



# Markov Decision Processes (MDPs)

Consists of:

- **S:** Set of states
- **A:** Set of actions
- **P(s' | s, a):** Transition probability from state s to s' after action a
- **R(s, a, s'):** Reward received after taking action a in state s



# Policies in RL

- Deterministic case: Policies map **states to actions**

$$\pi : S \rightarrow A$$

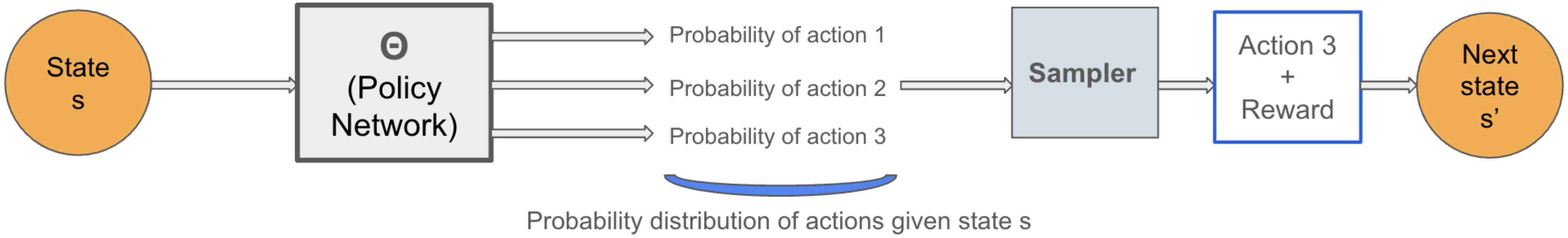
- Stochastic case: Policies map **states to distributions over actions**

$$\pi : S \rightarrow \mathcal{P}(A)$$

- **Goal:** find optimal policies that maximize reward

# Policy Networks

- Policy networks implement stochastic policies

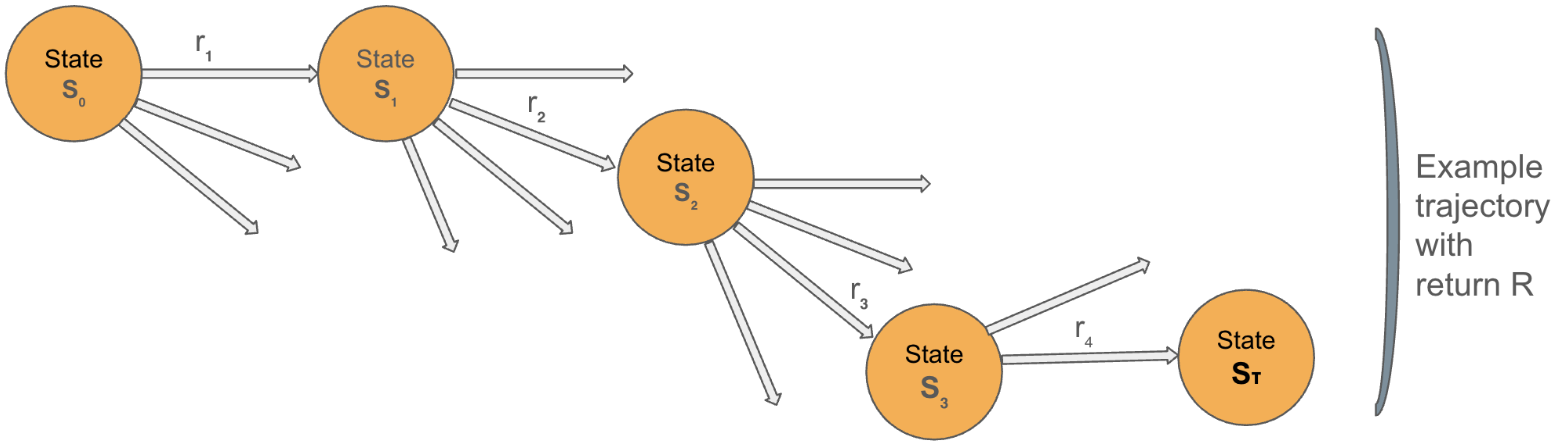



- Optimize policies by updating θ

# Training Policy Networks

- **How do we update Θ based on observed trajectories?**
  - This is similar to **supervised learning** methods

- Sample multi-hop trajectories for policy π and treat the sampled trajectories as training data



# Policy Objective Function

- Use **gradient ascent** to iteratively adjust policy parameters θ to maximize the expected return, **policy objective function J(θ)**

$$\max_{\theta} J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)] = \sum_{\boxed{\tau}} \underbrace{P(\tau;\theta)}_{\text{Probability of trajectory}} \underbrace{R(\tau)}_{\text{Cumulative discounted return from trajectory}}$$

All possible trajectories τ consistent with policy π

- Find the gradient of the policy objective function:

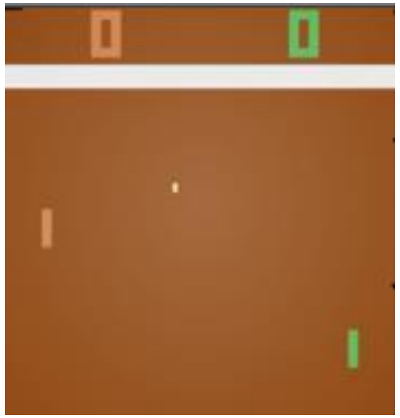
$$\nabla_\theta J(\theta) = \nabla_\theta \sum_{\tau} P(\tau;\theta) R(\tau)$$

- **Gradient ascent update,** where α is the step size:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

# MDPs in Pong

**State:**
210×160×3 RGB pixels → vector of D=80×80=6400 features



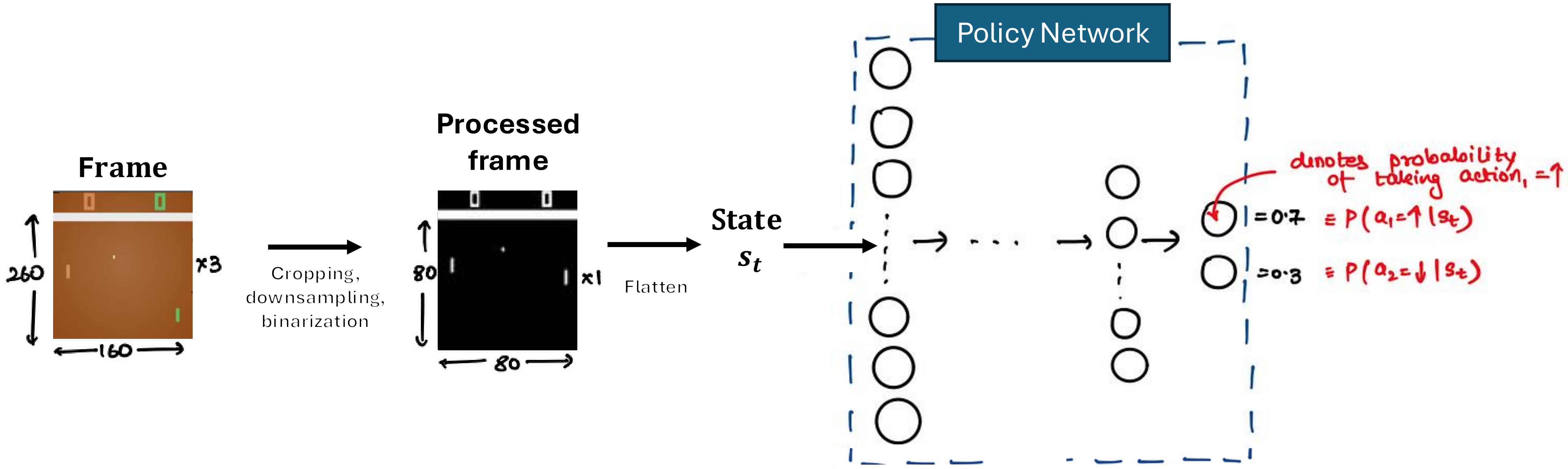
**Action Space:**
- UP
- DOWN

**Rewards (Sparse):**
- +1 if opponent misses ball
- -1 if agent misses ball
- Episode continues until one player scores 21 points

1) Sequence of frames → state representation
2) State representation → action

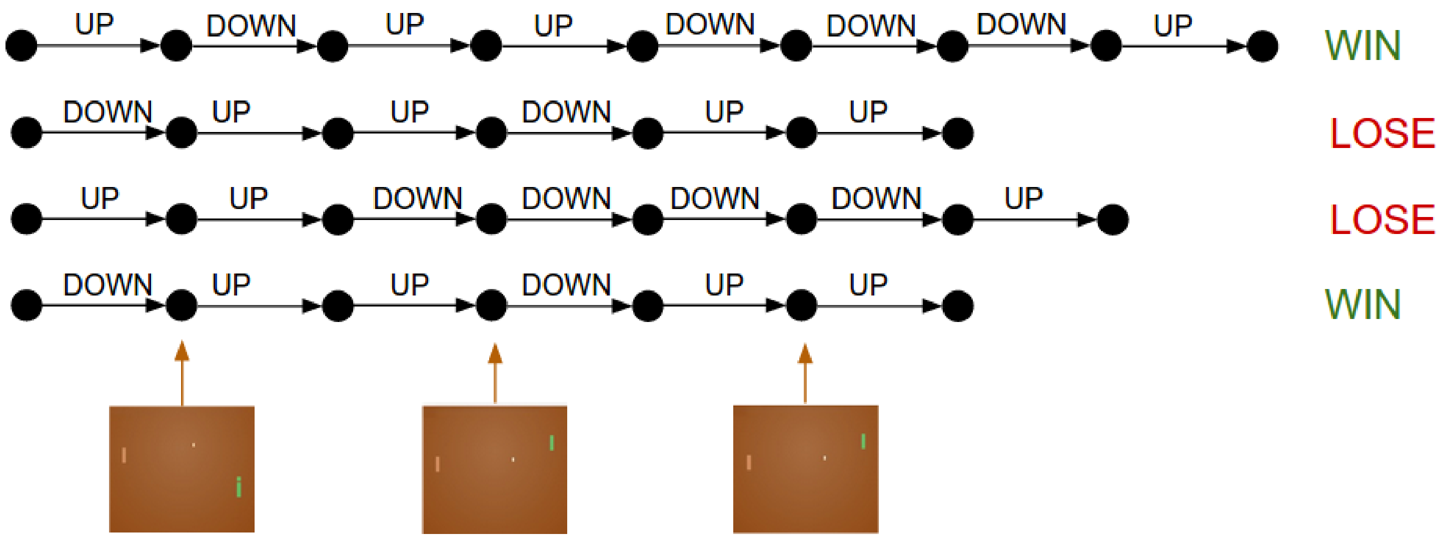
# Pong Policies

- Stochastic policy
- Two-layer fully connected multi-layer perceptron (MLP)



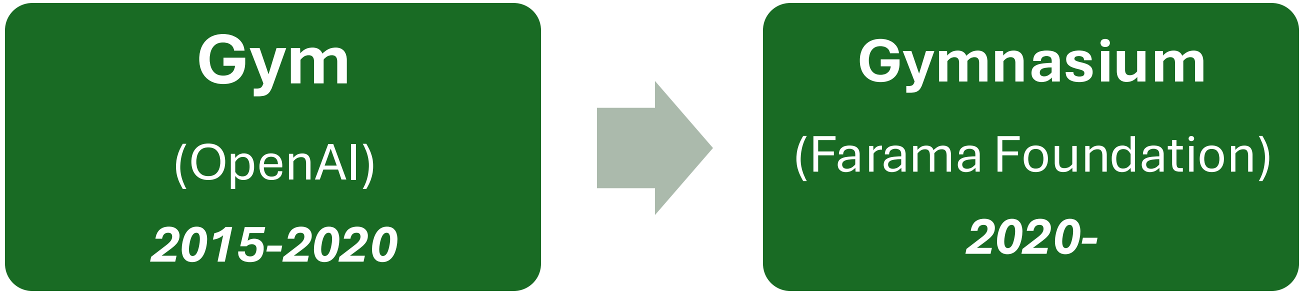
# Training Pong Policies

- Four game examples (episodes)
- **Input:** difference between the current preprocessed frame and the previous preprocessed frame, $x_t$=prepro($obs_t$)−prepro($obs_{t-1}$)
  - Ideally, we want to use more than 2 frames

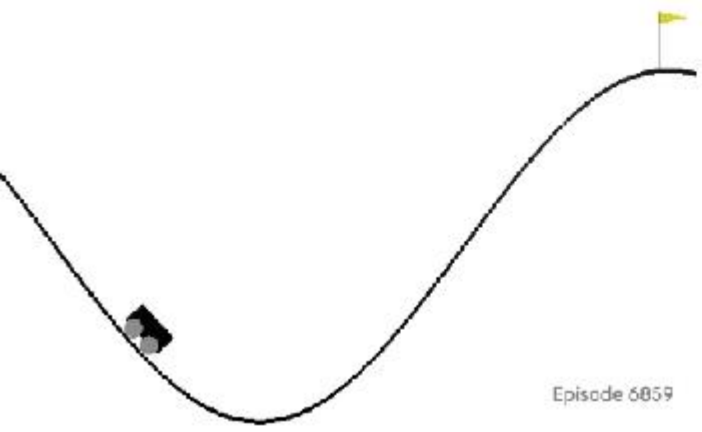


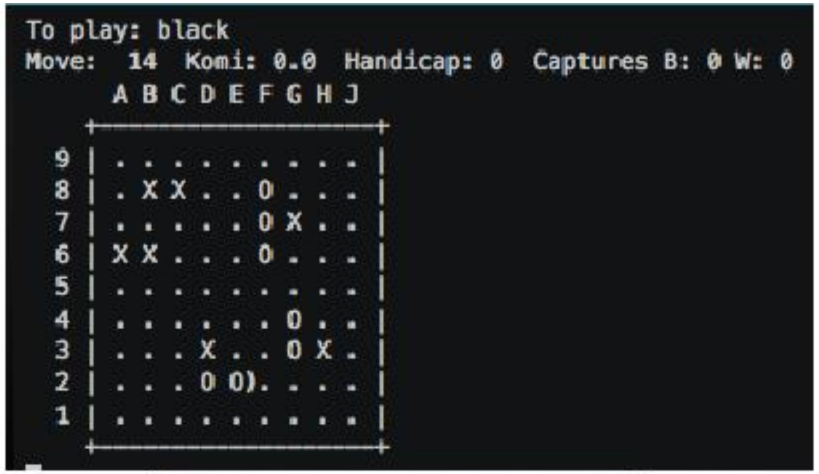
# Gym/Gymnasium

- A foundational API for RL environments
- OpenAI **Gym** dates to 2015
- Since 2020: **Gymnasium** (Farama Foundation)
  - Community-maintained successor to Gym; nearly identical usage
  - **Shimmy** can be used for cross-compatibility across environments
- Pre-defined environments
  - Atari suite, MuJoCo, etc…

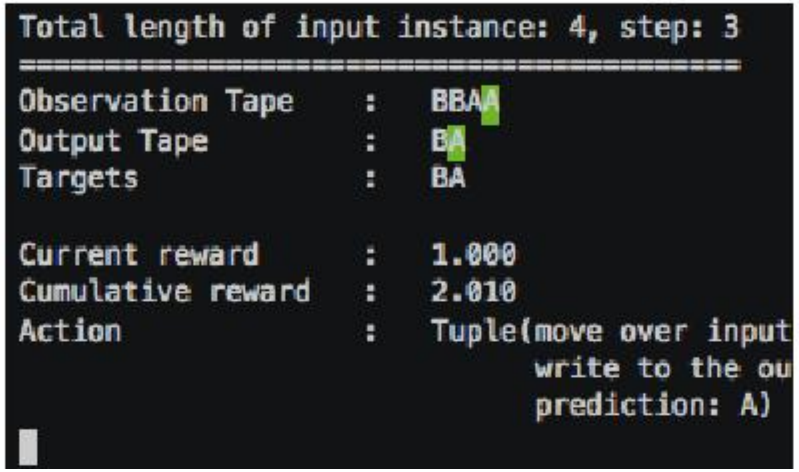


# Gym Environments

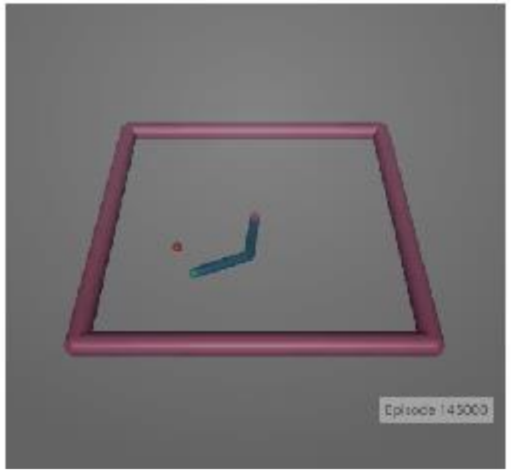








# Demo

- Link here