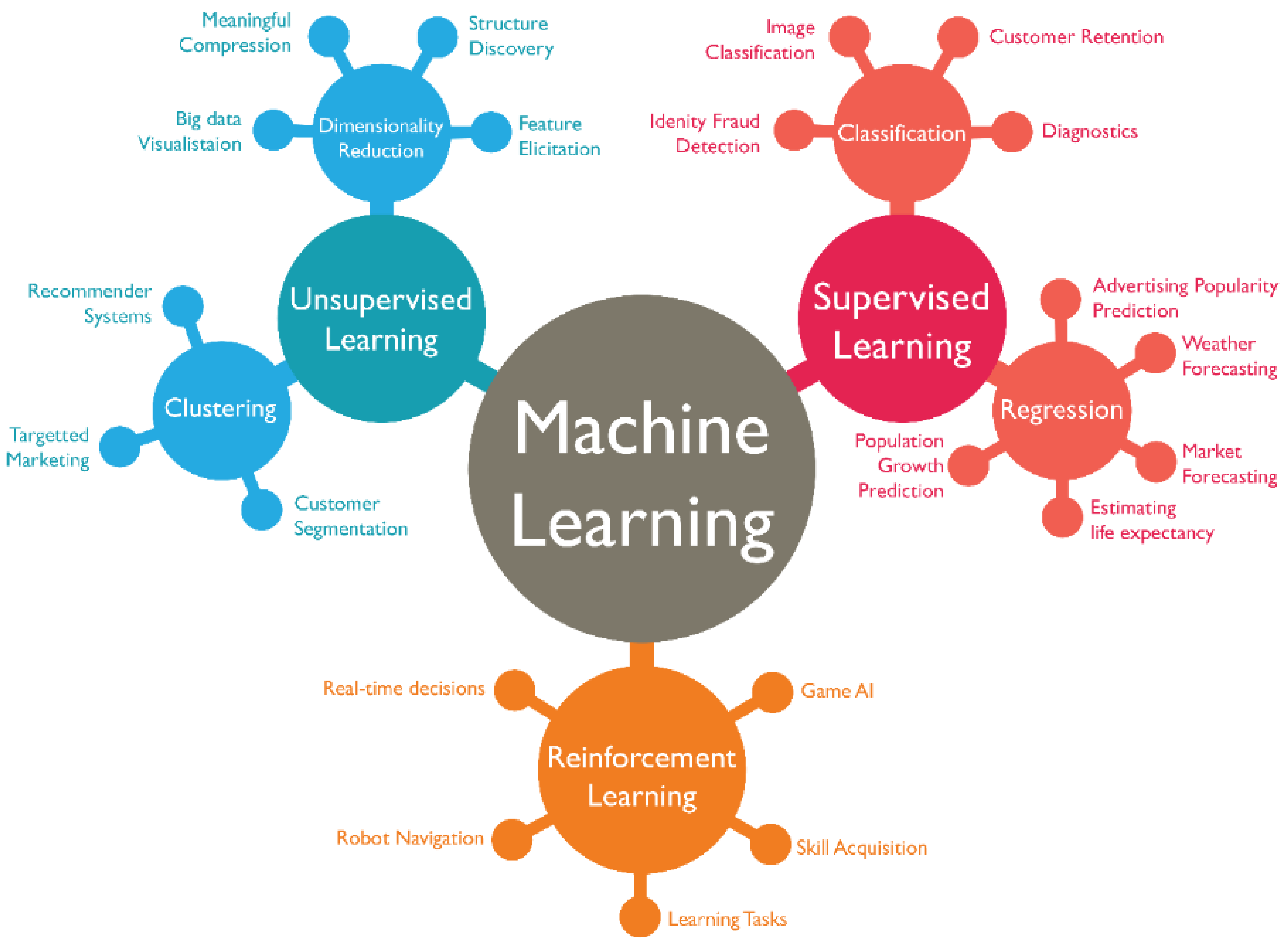
UNIVERSITY OF TECHNOLOGY
IN THE EUROPEAN CAPITAL OF CULTURE
CHEMNITZ

Deep Reinforcement Learning

Introduction

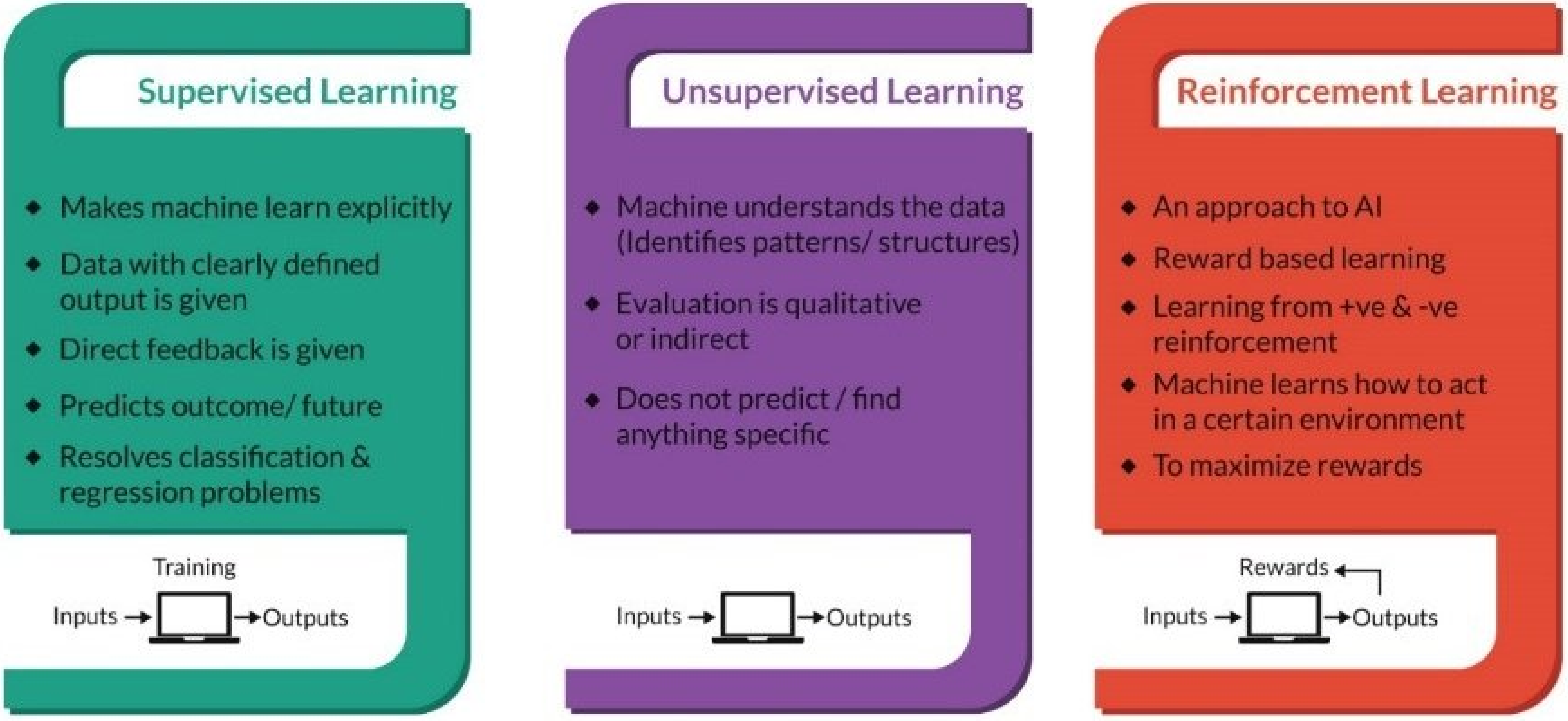
Julien Vitay
Professur für Künstliche Intelligenz - Fakultät für Informatik

1 - What is reinforcement learning?



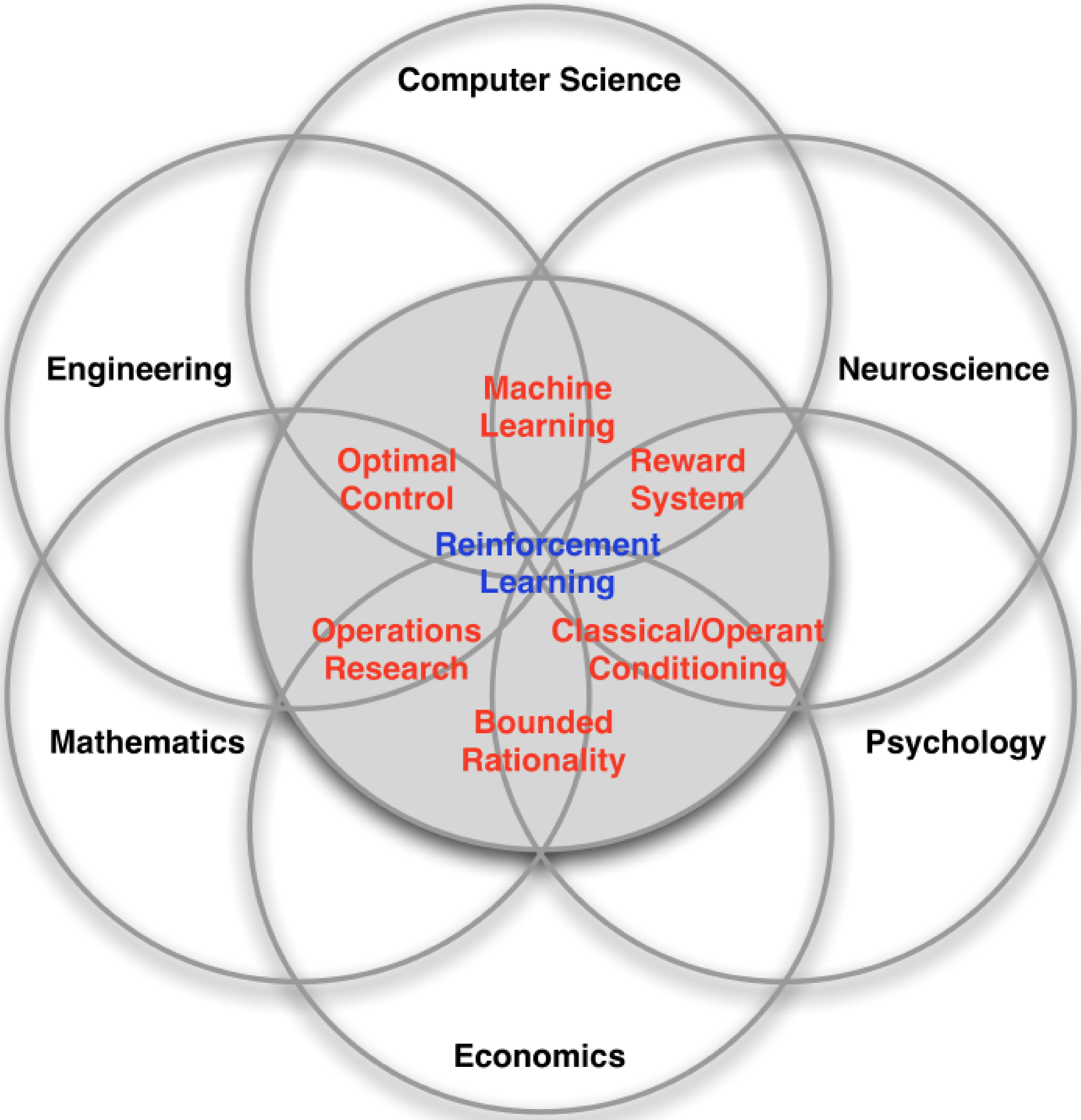
Different types of machine learning depending on the feedback

- **Supervised learning:** the correct answer is provided to the system.
- **Unsupervised learning:** no answer is given to the system.
- **Reinforcement learning:** an estimation of the correctness of the answer is provided.



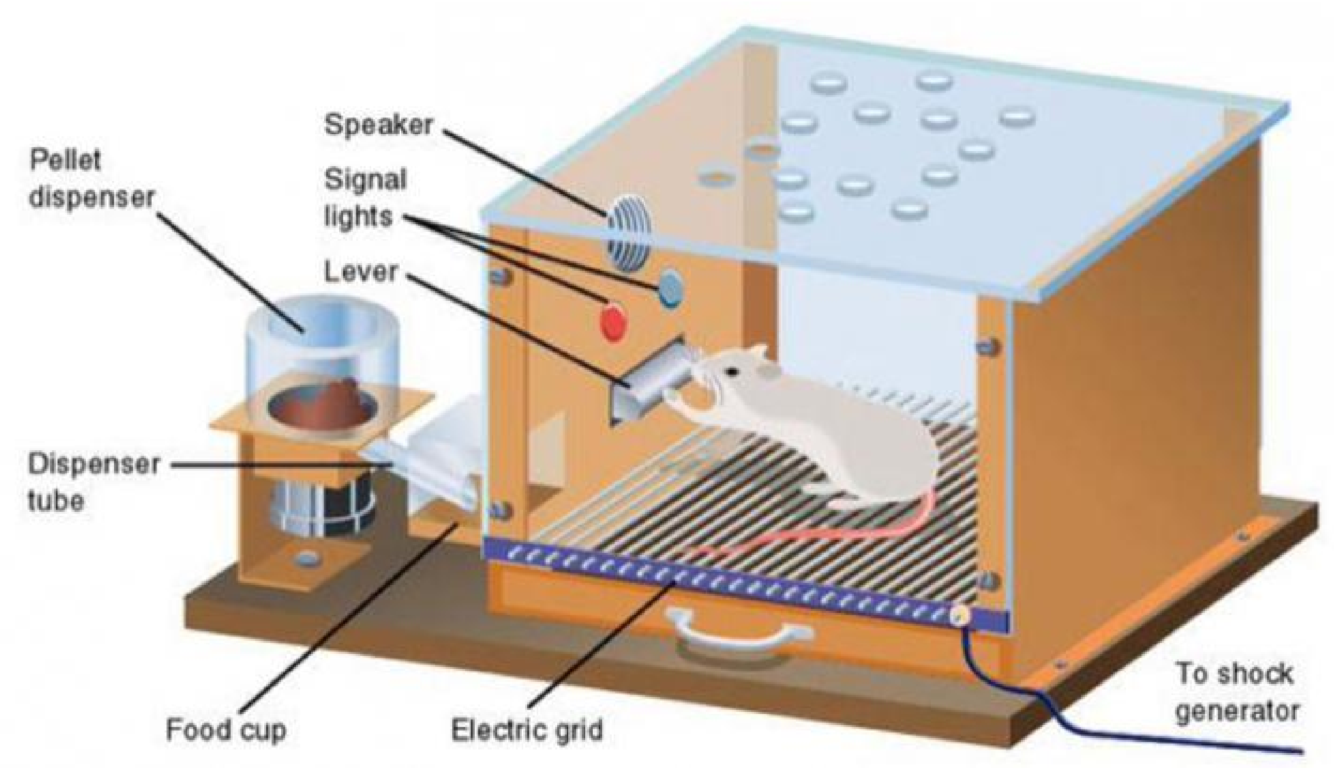
Source: https://www.analyticsvidhya.com/blog/2016/12/artificial-intelligence-demystified/

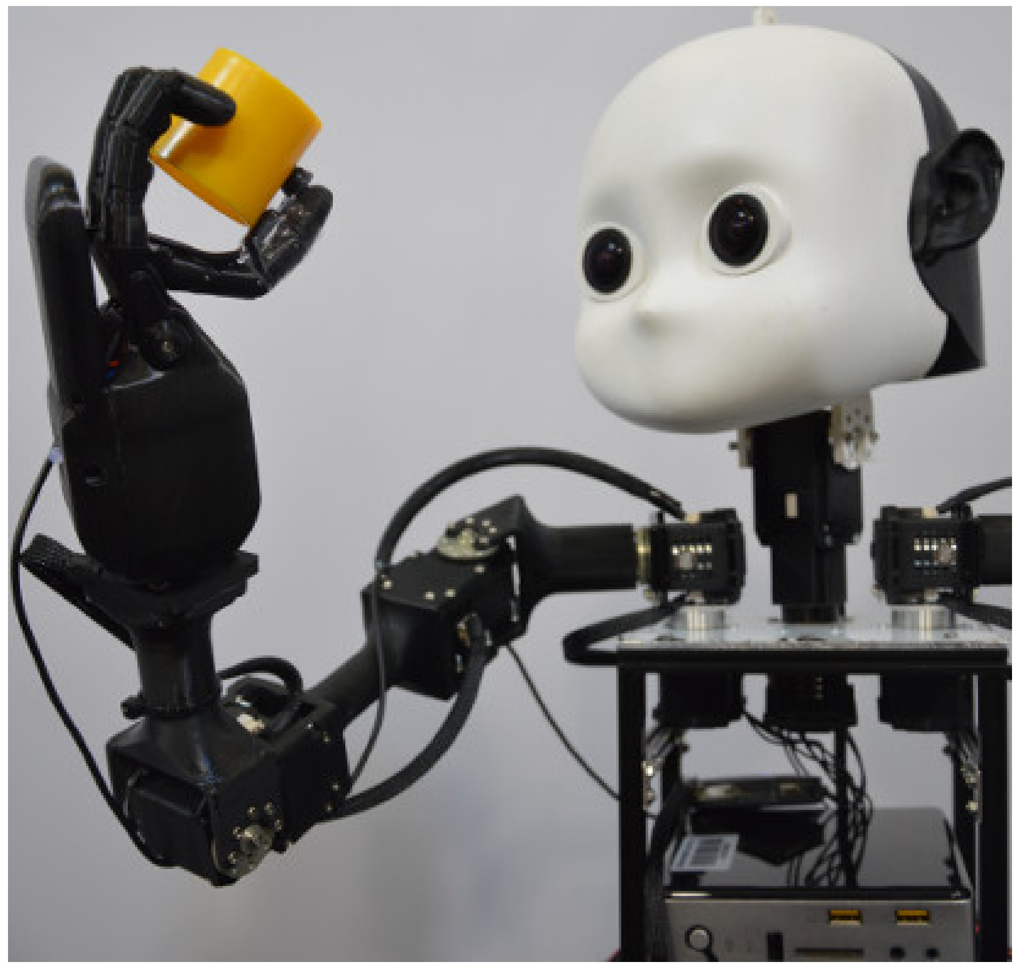
Many faces of RL



Source: David Silver. http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html

A brief history of reinforcement learning

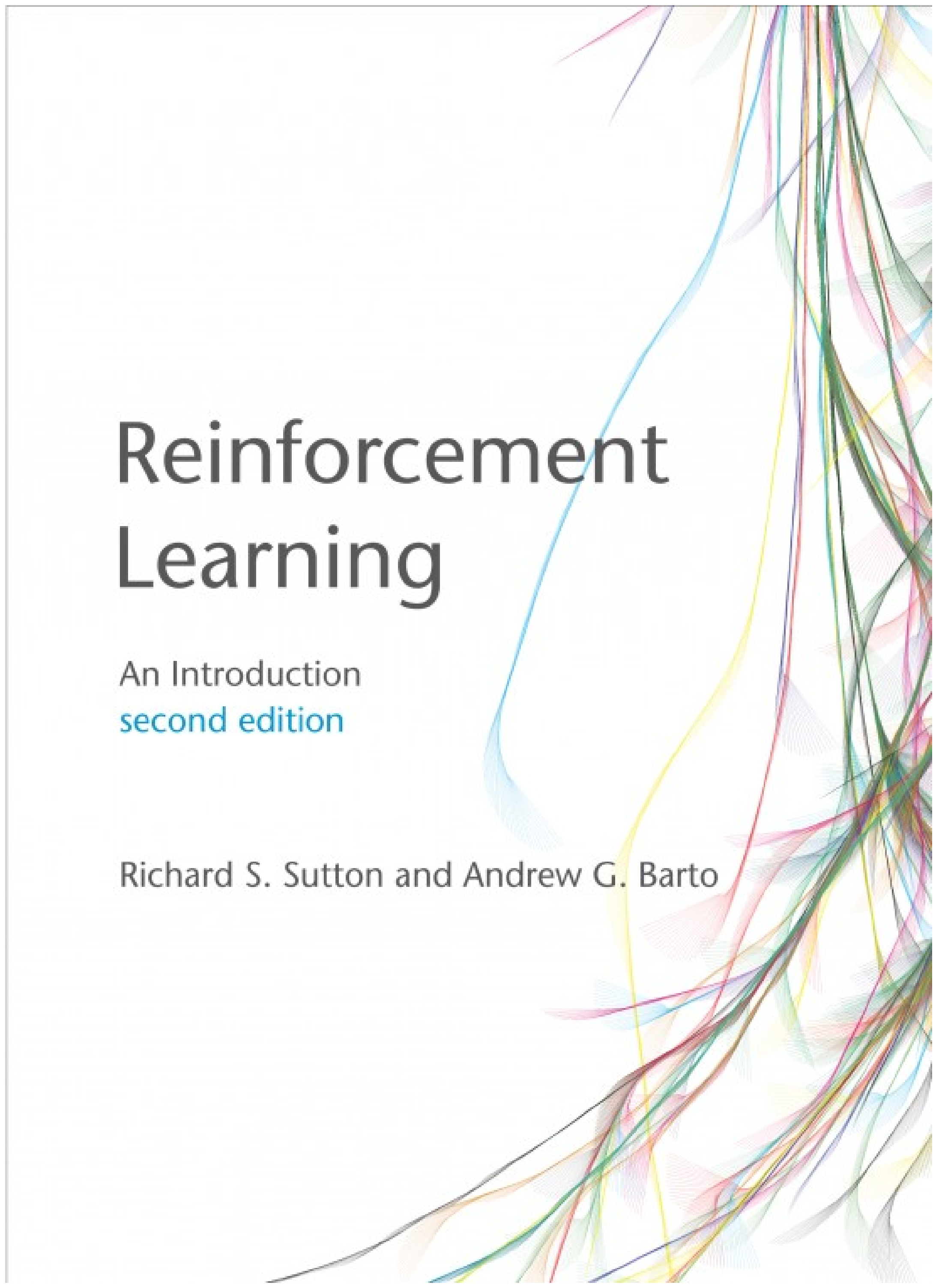




- **Early 20th century**: animal behavior, psychology, operant conditioning
  - Ivan Pavlov, Edward Thorndike, B.F. Skinner
- **1950s**: optimal control, Markov Decision Process, dynamic programming
  - Richard Bellman, Ronald Howard
- **1970s**: trial-and-error learning
  - Marvin Minsky, Harry Klopf, Robert Rescorla, Allan Wagner
- **1980s**: temporal difference learning, Q-learning
  - Richard Sutton, Andrew Barto, Christopher Watkins, Peter Dayan
- **2013-now**: deep reinforcement learning
  - Deepmind (Mnih, Silver, Graves, Hassabis...)
  - OpenAI (Sutskever, Schulman...)
  - Berkeley (Sergey Levine)

See http://www.incompleteideas.net/book/ebook/node12.html

The RL bible



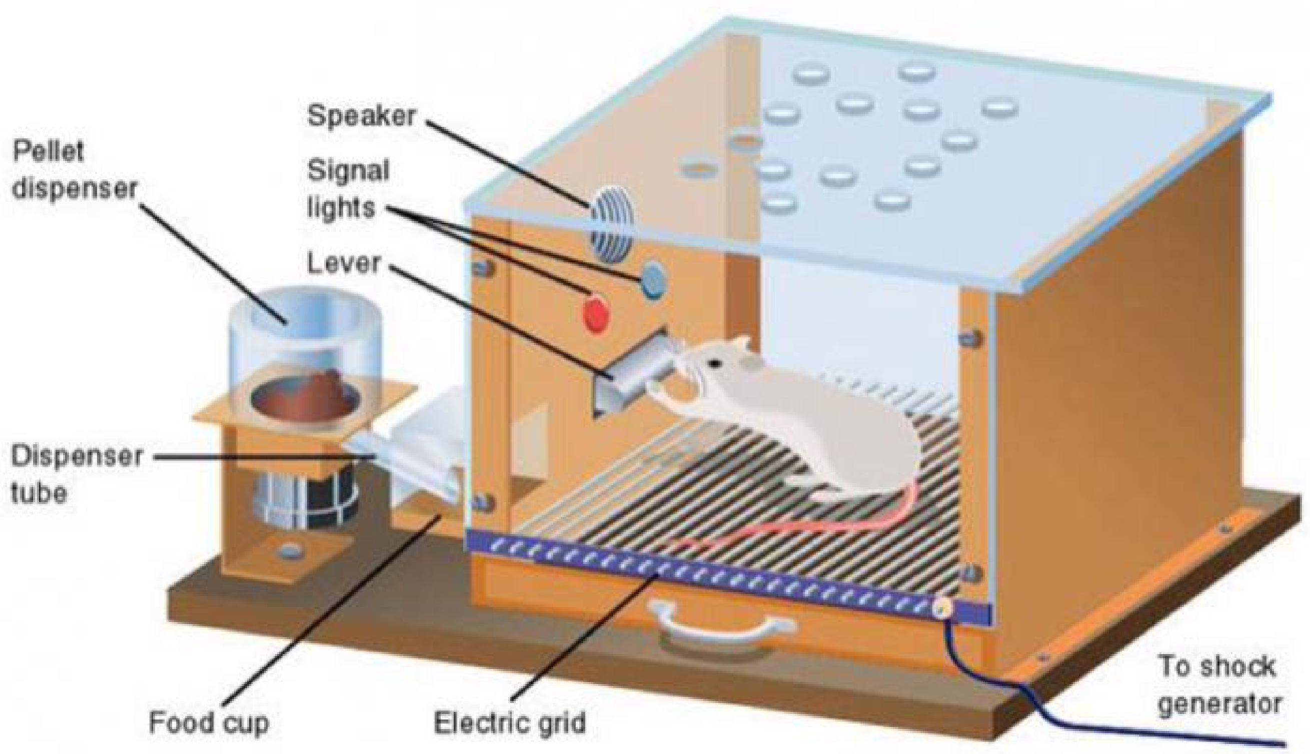
Sutton and Barto (1998) . Reinforcement Learning: An Introduction. MIT Press.

Sutton and Barto (2017) . Reinforcement Learning: An Introduction. MIT Press. 2nd edition.

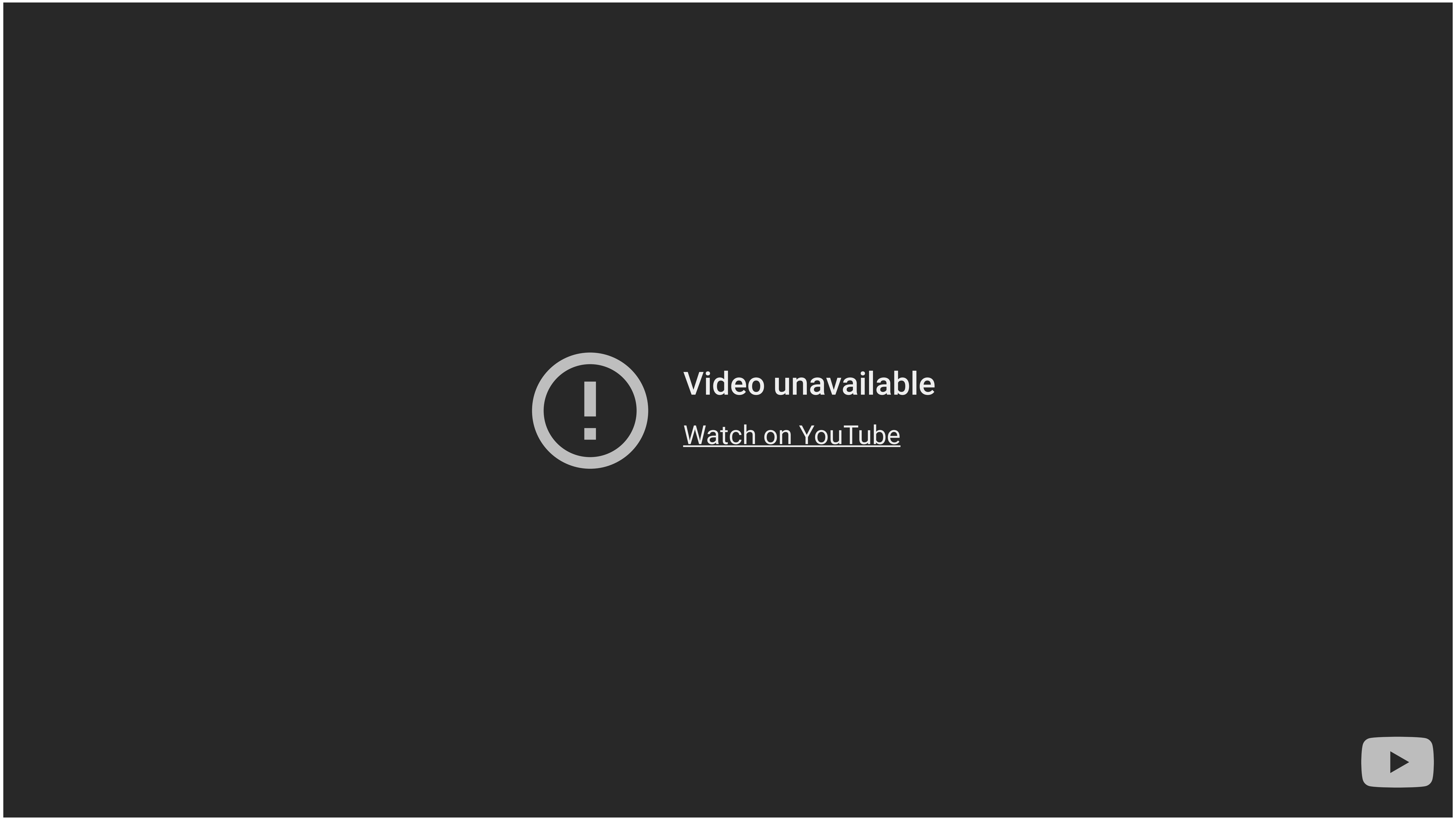
http://incompleteideas.net/sutton/book/the-book.html

Operant conditioning

- Reinforcement learning comes from animal behavior studies, especially **operant conditioning / instrumental learning**.
- **Thorndike's Law of Effect** (1874–1949) suggested that behaviors followed by satisfying consequences tend to be repeated and those that produce unpleasant consequences are less likely to be repeated.
- Positive reinforcements (**rewards**) or negative reinforcements (**punishments**) can be used to modify behavior (**Skinner's box, 1936**).
- This form of learning applies to all animals, including humans:
  - Training (animals, children…)
  - Addiction, economics, gambling, psychological manipulation…
- **Behaviorism:** only behavior matters, not mental states.



Operant conditioning



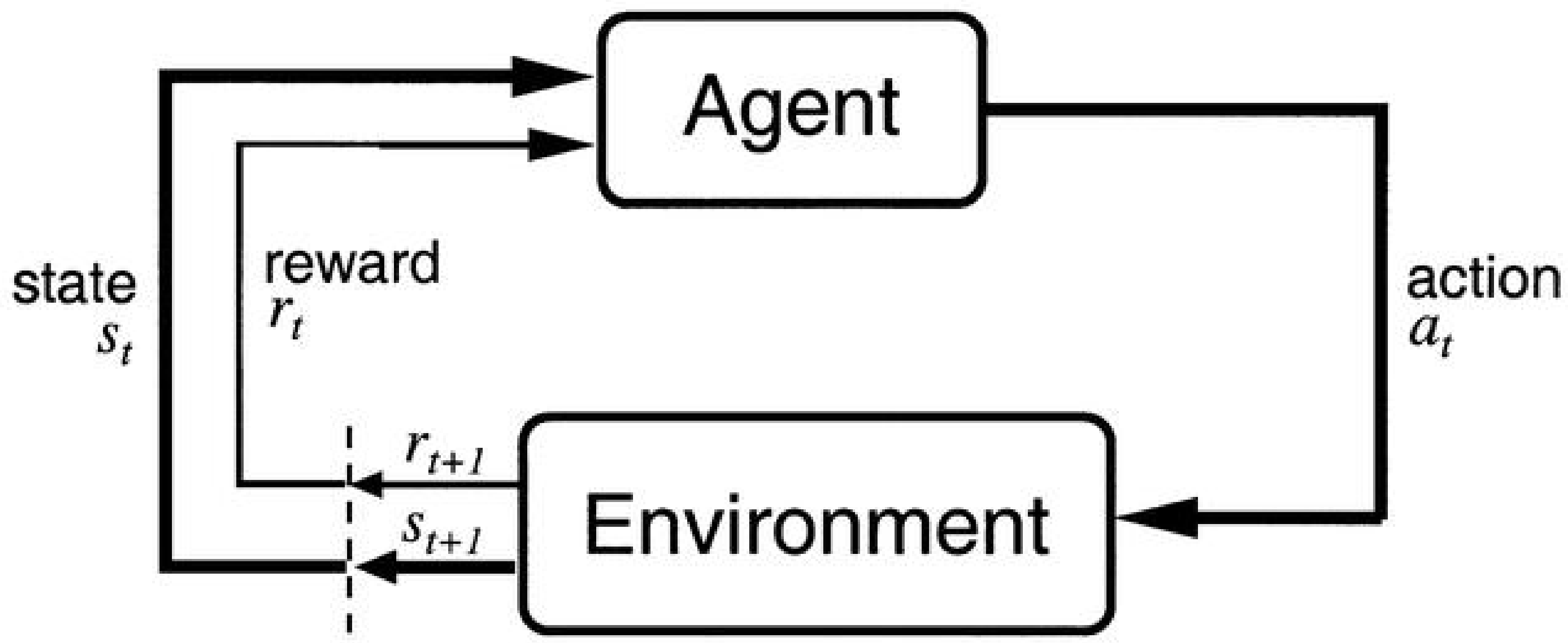
Trial and error learning

- The key concept of RL is **trial and error** learning.
- The agent (rat, robot, algorithm) tries out an **action** and observes the **outcome**.
    - If the outcome is positive (reward), the action is reinforced (more likely to occur again).
    - If the outcome is negative (punishment), the action will be avoided.
- After enough interactions, the agent has **learned** which action to perform in a given situation.

Trial and error learning

- RL is merely a formalization of the trial-and-error learning paradigm.
- The agent has to **explore** its environment via trial-and-error in order to gain knowledge.
- The biggest issue with this approach is that exploring large action spaces might necessitate a **lot** of trials (**sample complexity**).
- The modern techniques we will see in this course try to reduce the sample complexity.

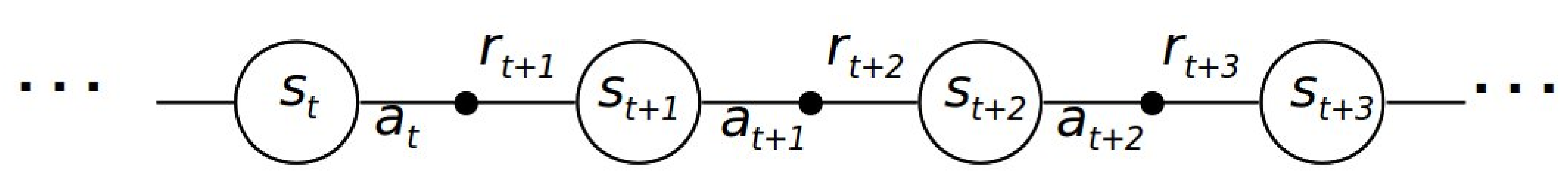
The agent-environment interface



Source: Sutton and Barto (1998).

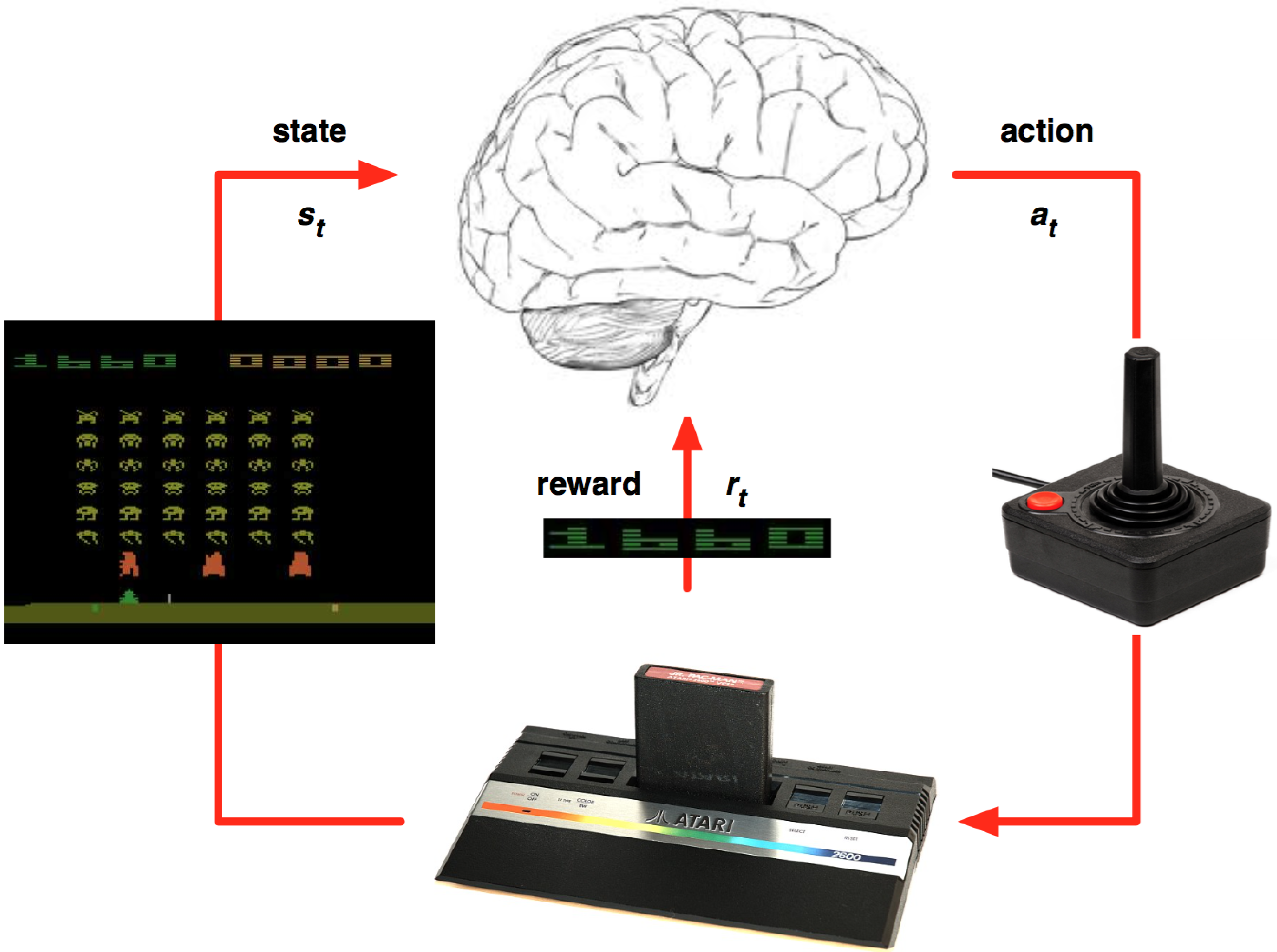
- The agent and the environment interact at discrete time steps: t=0, 1, ...
- The agent observes its state at time t: $s_t \in \mathcal{S}$
- It produces an action at time t, depending on the available actions in the current state: $a_t \in \mathcal{A}(s_t)$
- It receives a reward according to this action at time t+1: $r_{t+1} \in \Re$
- It updates its state: $s_{t+1} \in \mathcal{S}$

- The behavior of the agent is therefore is a sequence of **state-action-reward-state** (s, a, r, s') transitions.



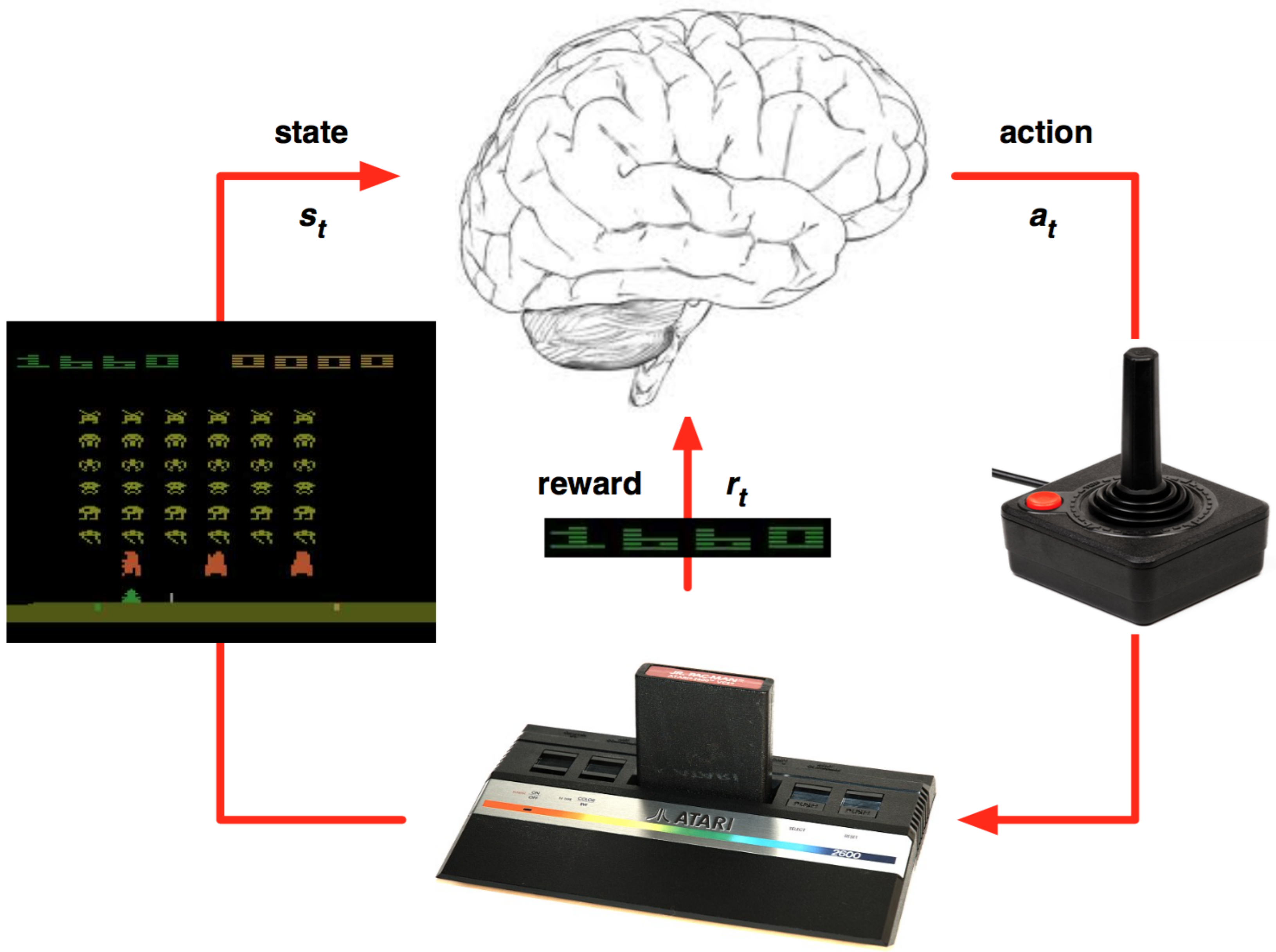
- Sequences $\tau = (s_0, a_0, r_1, s_1, a_1, \ldots, s_T)$ are called **episodes**, **trajectories**, **histories** or **rollouts**.

The agent-environment interface



Source: David Silver. http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html

Environment and agent states



Source: David Silver.
http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html

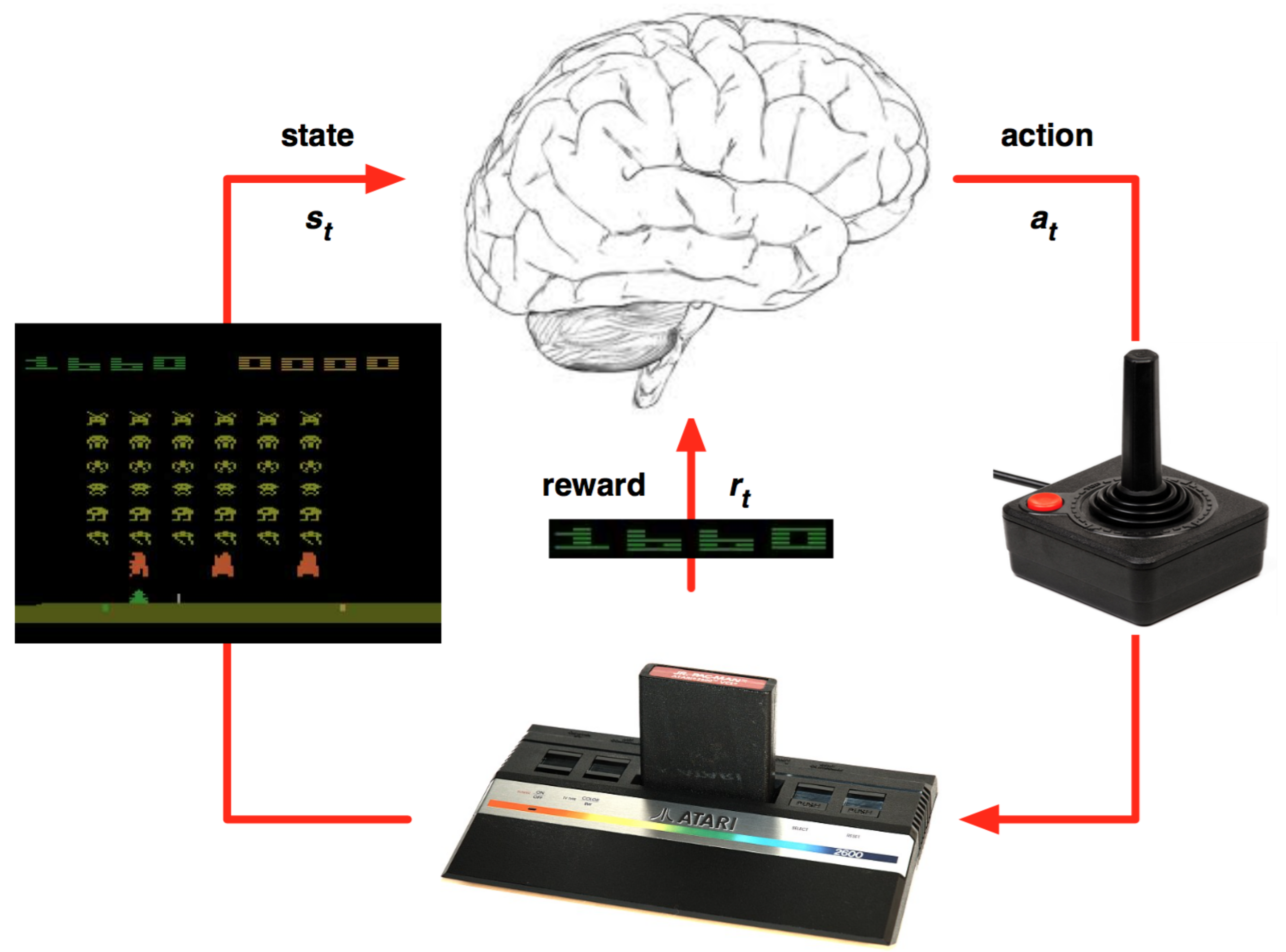
- The state s_t can relate to:
  - the **environment state**, i.e. all information external to the agent (position of objects, other agents, etc).
  - the **internal state**, information about the agent itself (needs, joint positions, etc).
- Generally, the state represents all the information necessary to solve the task.
- The agent generally has no access to the states directly, but to **observations** o_t:

$$o_t = f(s_t)$$

- Example: camera inputs do not contain all the necessary information such as the agent's position.
- Imperfect information define **partially observable problems**.

Policy

- What we search in RL is the optimal **policy**: which action a should the agent perform in a state s?
- The policy π maps states into actions.



Source: David Silver.
http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html

- It is defined as a **probability distribution** over states and actions:

$$\pi : \mathcal{S} \times \mathcal{A} \rightarrow P(\mathcal{S})$$
$$(s, a) \rightarrow \pi(s, a) = P(a_t = a | s_t = s)$$

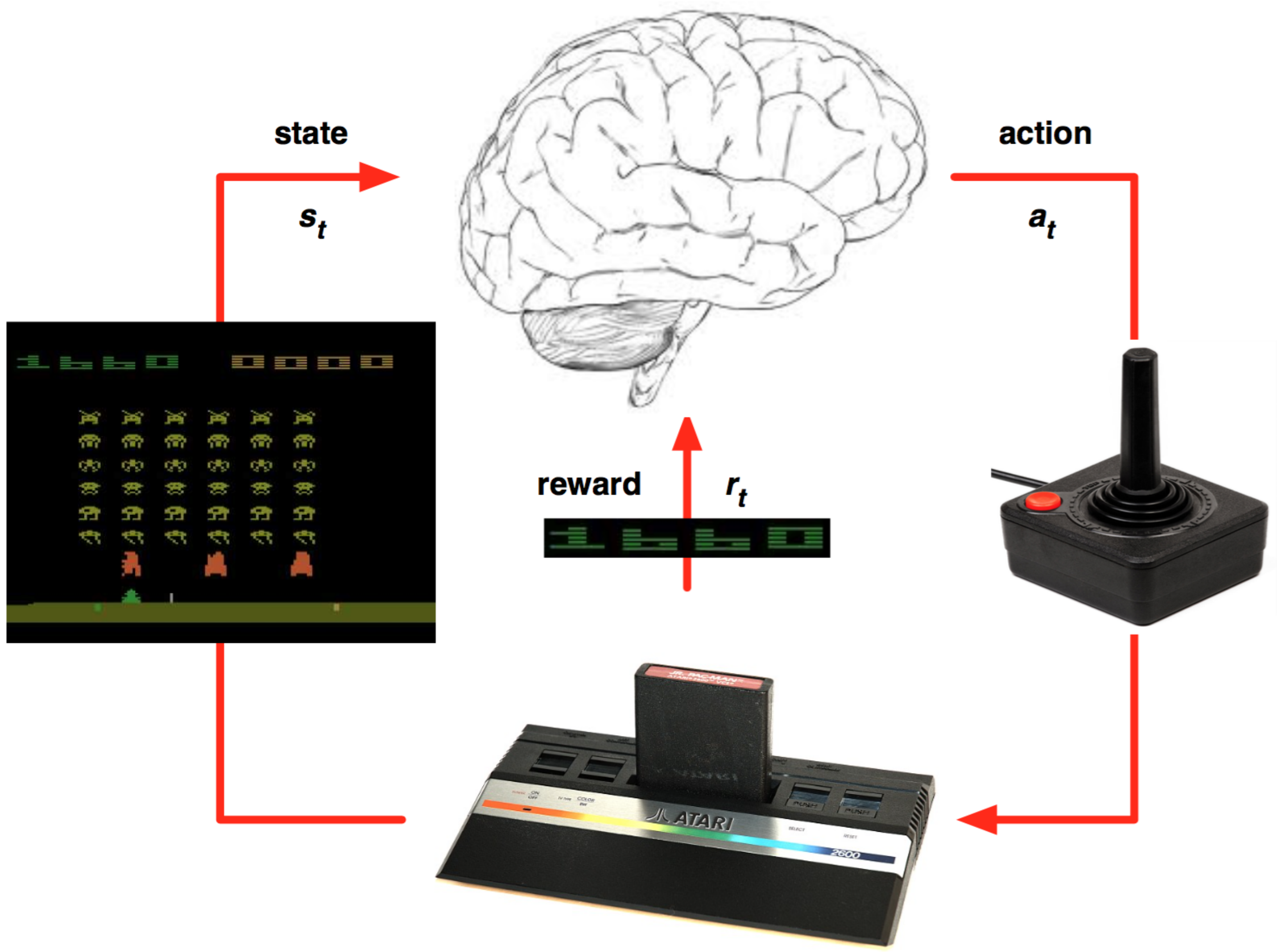
- $\pi(s, a)$ is the probability of selecting the action a in s. We have of course:

$$\sum_{a \in \mathcal{A}(s)} \pi(s, a) = 1$$

- Policies can be **probabilistic / stochastic**. **Deterministic policies** select a single action a^* in s:

$$\pi(s, a) = \begin{cases} 1 \text{ if } a = a^* \\ 0 \text{ if } a \neq a^* \end{cases}$$

Reward function



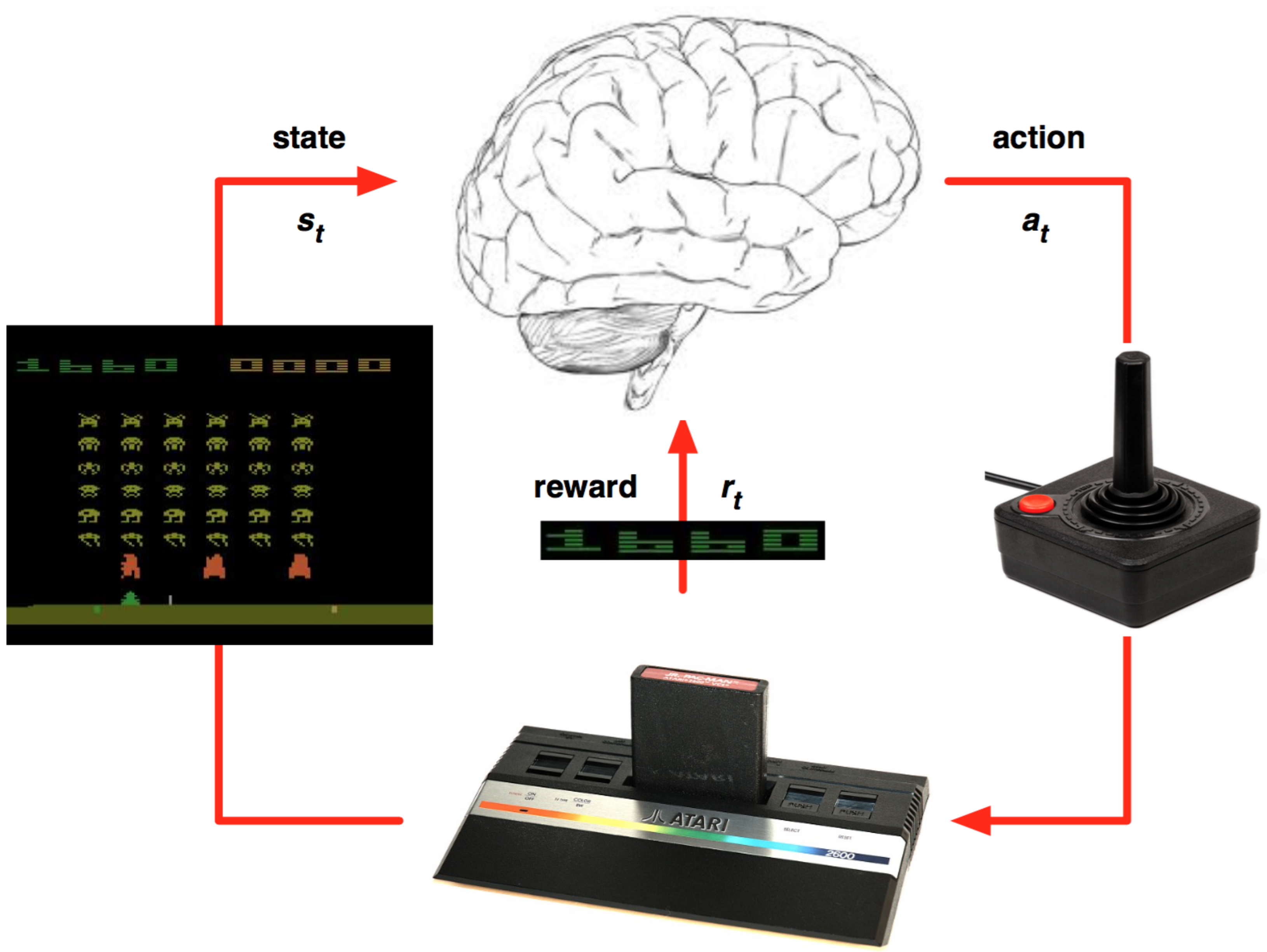
Source: David Silver.
http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html

- The only teaching signal in RL is the **reward function**.
- The reward is a scalar value r_{t+1} provided to the system after each transition (s_t, a_t, s_{t+1}).
- Rewards can also be probabilistic (casino).
- The mathematical expectation of these rewards defines the **expected reward** of a transition:

$$r(s, a, s') = \mathbb{E}_t [r_{t+1} | s_t = s, a_t = a, s_{t+1} = s']$$

- Rewards can be:
  - **dense**: a non-zero value is provided after each time step (easy).
  - **sparse**: non-zero rewards are given very seldom (difficult).

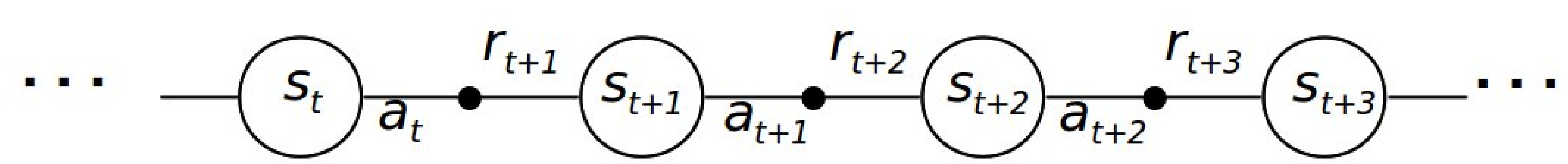
Returns



Source: David Silver.
http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html

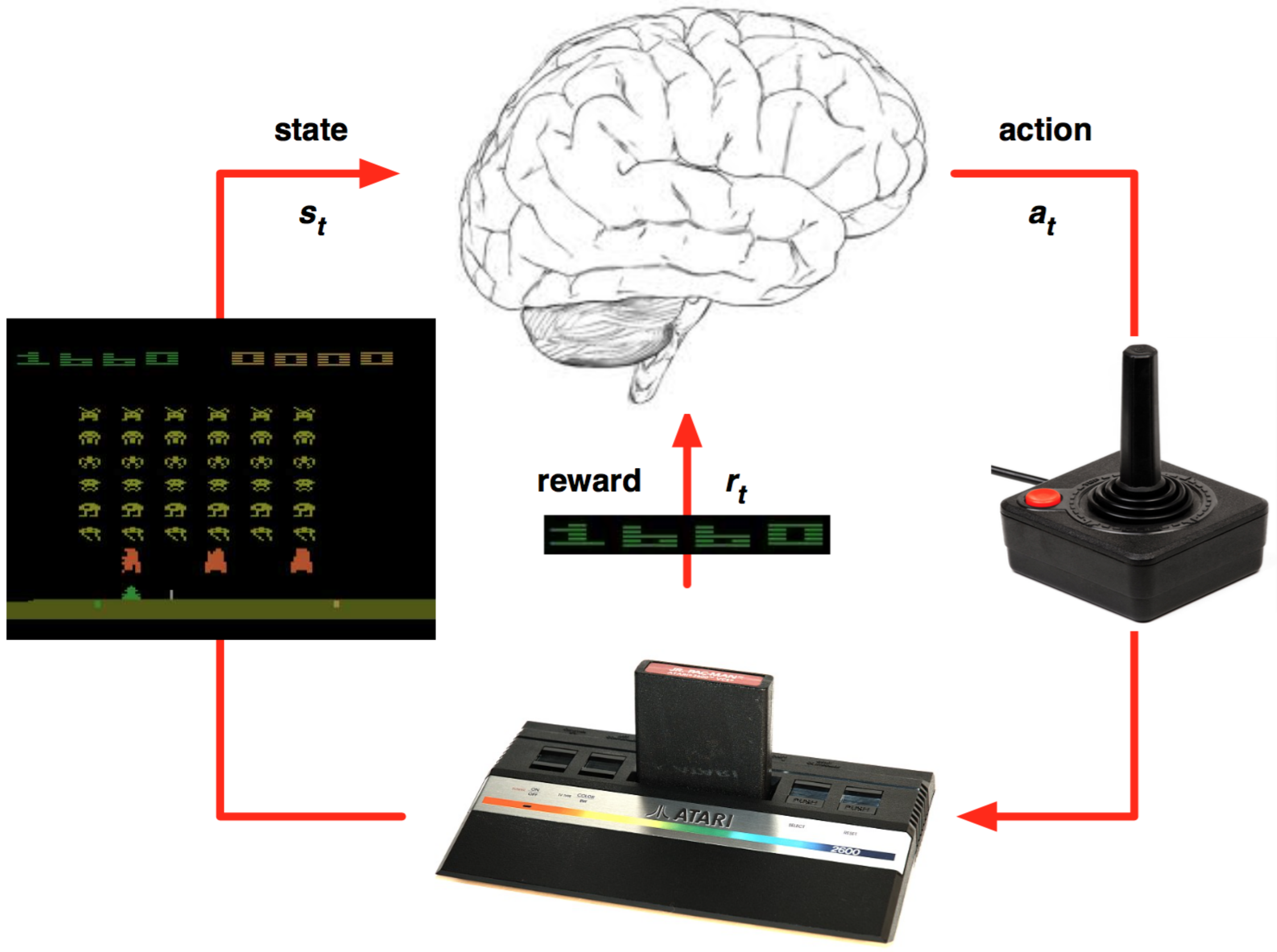
- The goal of the agent is to find a policy that **maximizes** the sum of future rewards at each timestep.
- The discounted sum of future rewards is called the **return**:

$$R_t = \sum_{k=0}^{\infty} \gamma^k \, r_{t+k+1}$$



- Rewards can be delayed w.r.t to an action: we care about all future rewards to select an action, not only the immediate ones.
- Example: in chess, the first moves are as important as the last ones in order to win, but they do not receive reward.

Value functions



Source: David Silver.
http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html

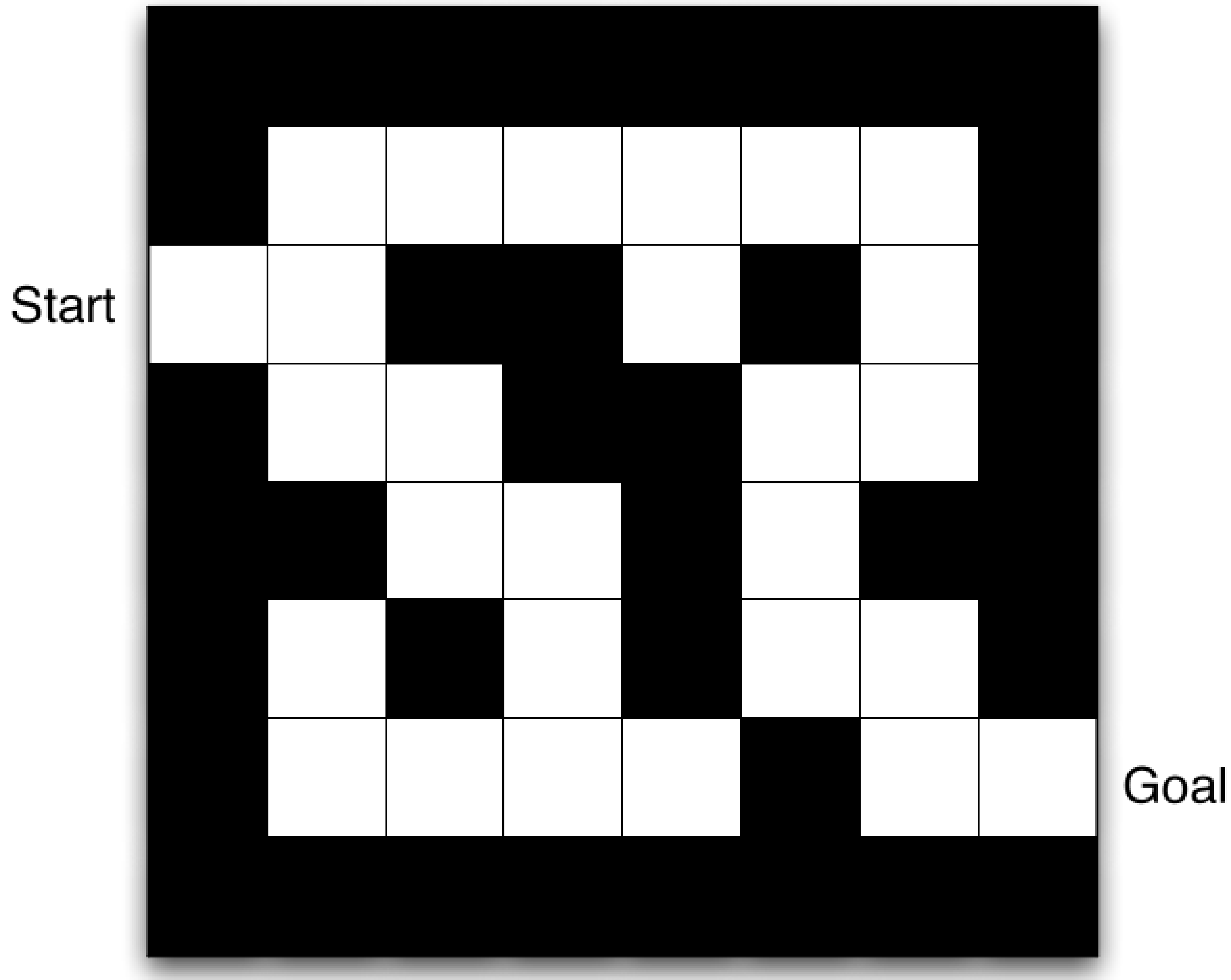
- The **expected return** in a state s is called its **value**:

$$V^{\pi}(s) = \mathbb{E}_{\pi}(R_t | s_t = s)$$

- The value of a state defines how good it is to be in that state.
- If a state has a high value, it means we will be able to collect a lot of rewards **on the long term** and **on average**.

- Value functions are central to RL: if we know the value of all states, we can infer the policy.
- The optimal action is the one that leads to the state with the highest value.
- Most RL methods deal with estimating the value function from experience (trial and error).

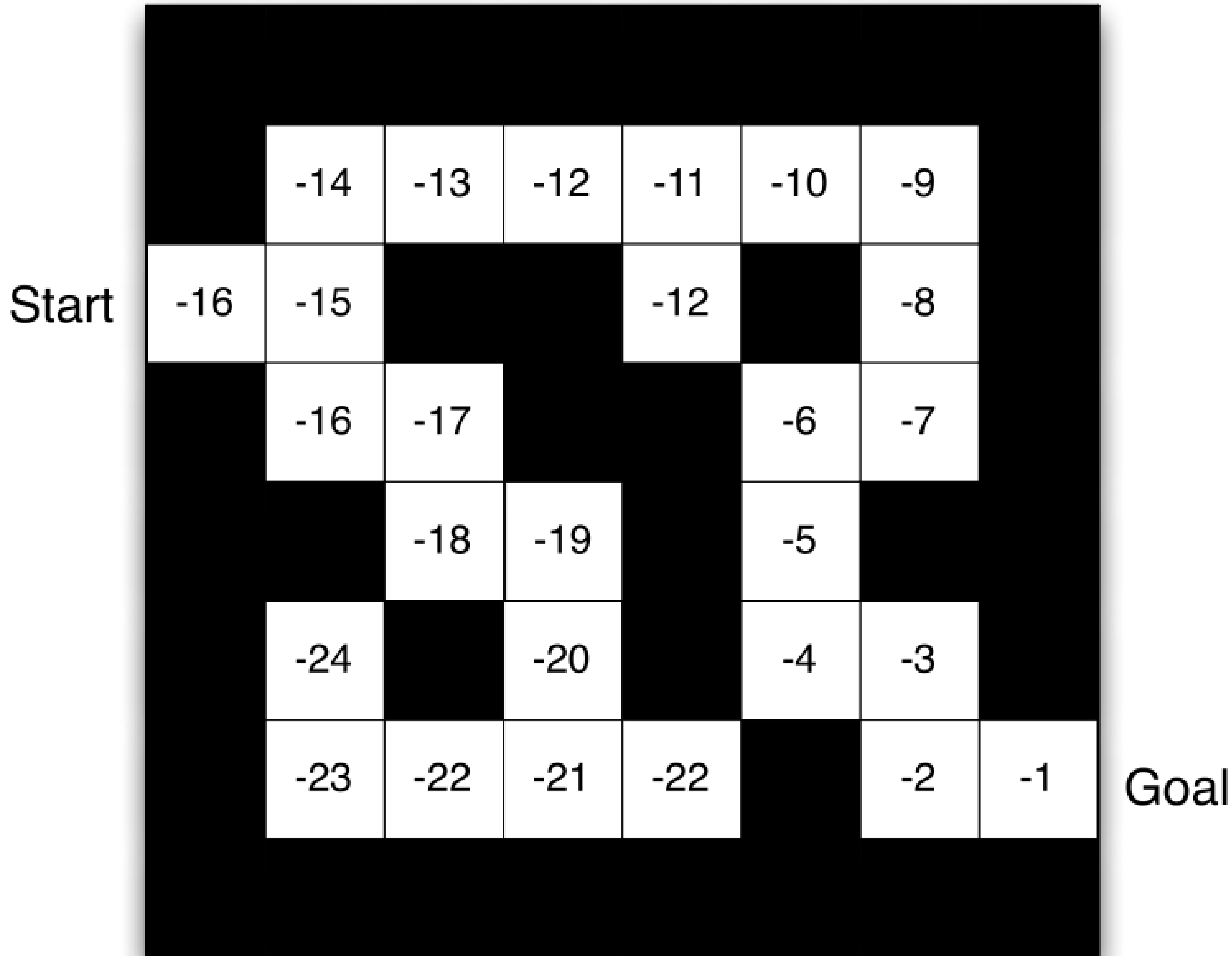
Simple maze



Goal: finding the exit as soon as possible.

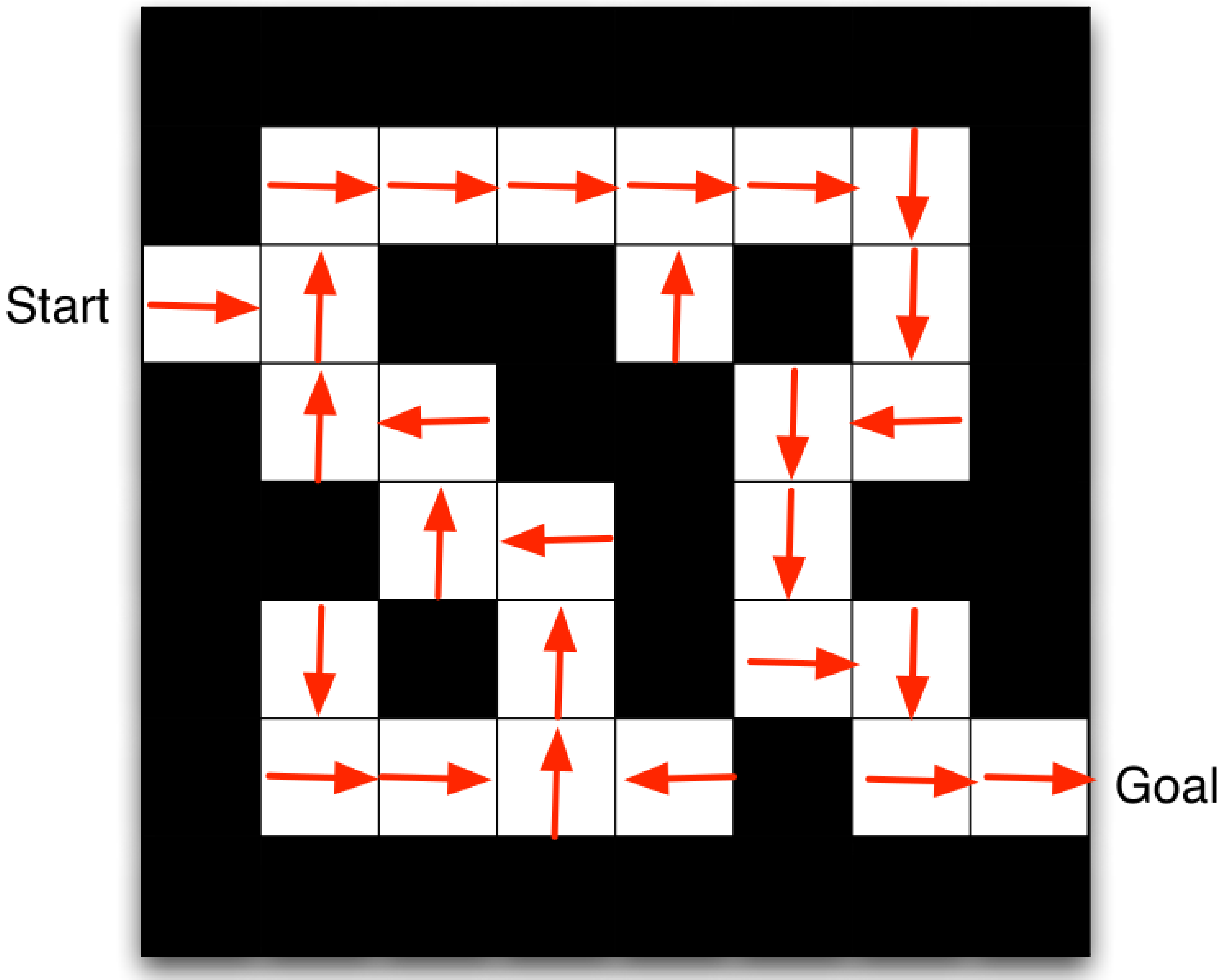
- **States**: position in the maze (1, 2, 3…).
- **Actions**: up, down, left, right.
- **Rewards**: -1 for each step until the exit.

Simple maze



- The value of each state indicates how good it is to be in that state.
- It can be learned by trial-and-error given a policy.

Simple maze

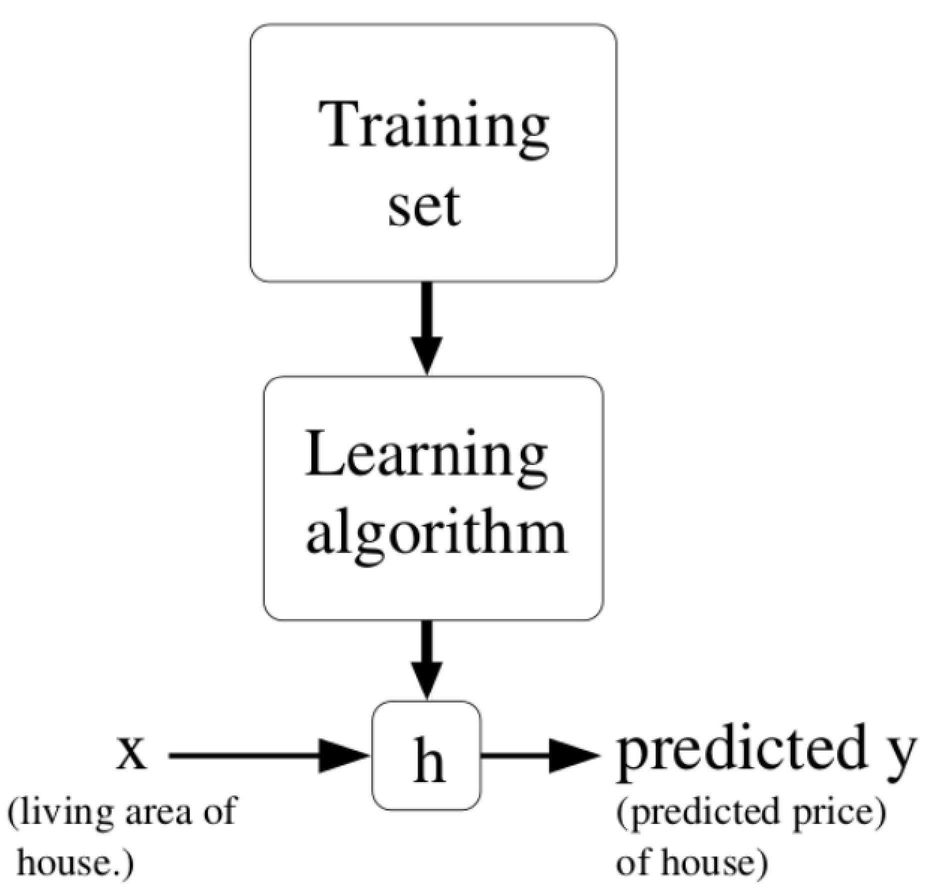


- When the value of all states is known, we can infer the optimal policy by choosing actions leading to the states with the highest value.

Note: the story is actually much more complicated…

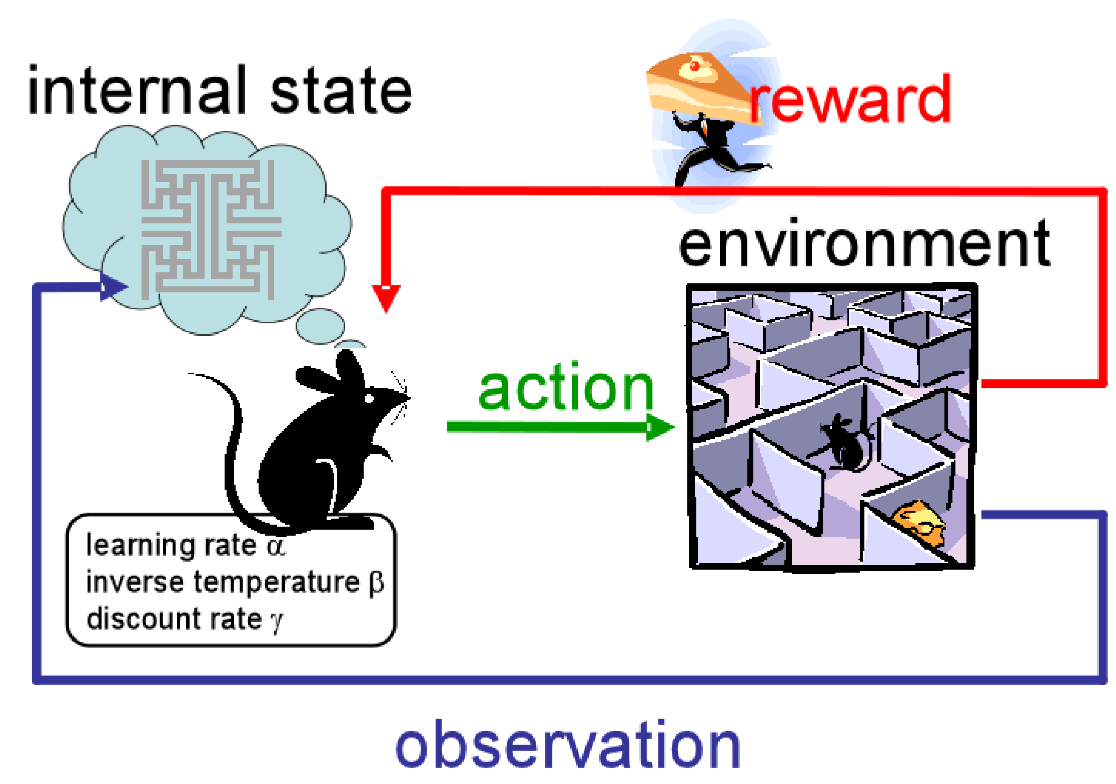
Supervised learning

- Correct input/output samples are provided by a **superviser** (training set).
- Learning is driven by **prediction errors**, the difference between the prediction and the target.
- Feedback is **instantaneous**: the target is immediately known.
- **Time** does not matter: training samples are randomly sampled from the training set.



Reinforcement learning

- Behavior is acquired through **trial and error**, no supervision.
- **Reinforcements** (rewards or punishments) change the probability of selecting particular actions.
- Feedback is **delayed**: which action caused the reward? Credit assignment.
- **Time** matters: as behavior gets better, the observed data changes.

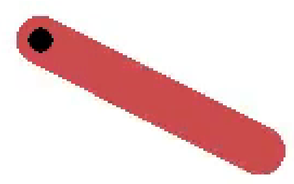


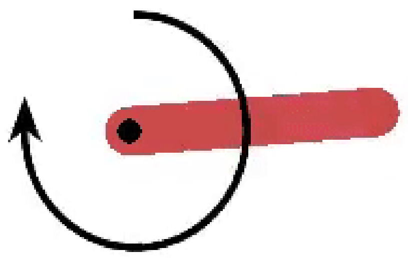
2 - Applications of RL

Optimal control

Pendulum

Goal: maintaining the pendulum vertical.





- **States**: angle and velocity of the pendulum.
- **Actions**: left and right torques.
- **Rewards**: cosine distance to the vertical.

Source: https://keras.io/examples/rl/ddpg_pendulum/

Optimal control

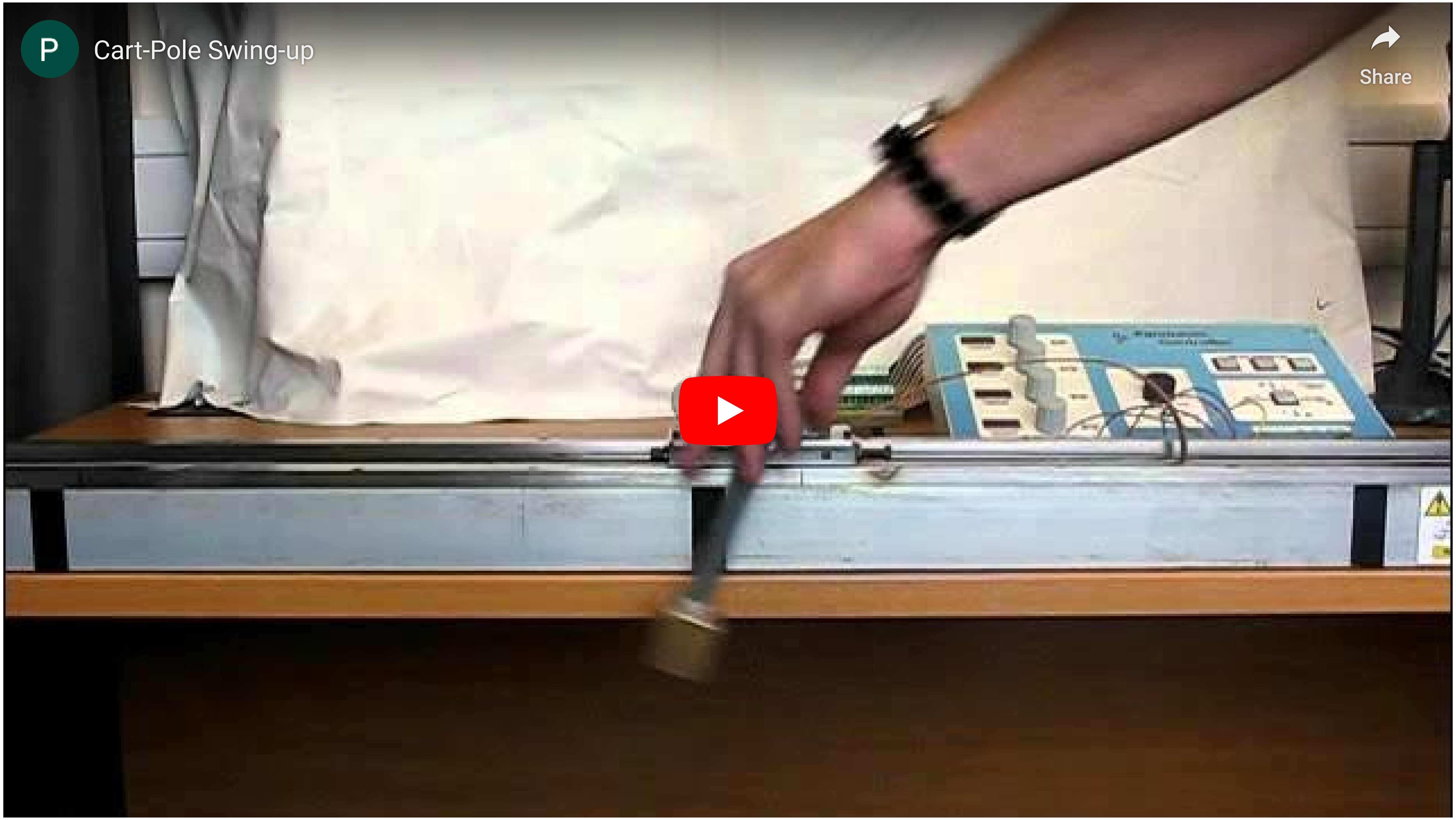
Cartpole

Goal: maintaining the pole vertical by moving the cart left or right.

- **States**: position and speed of the cart, angle and velocity of the pole.
- **Actions**: left and right movements.
- **Rewards**: +1 for each step until failure.

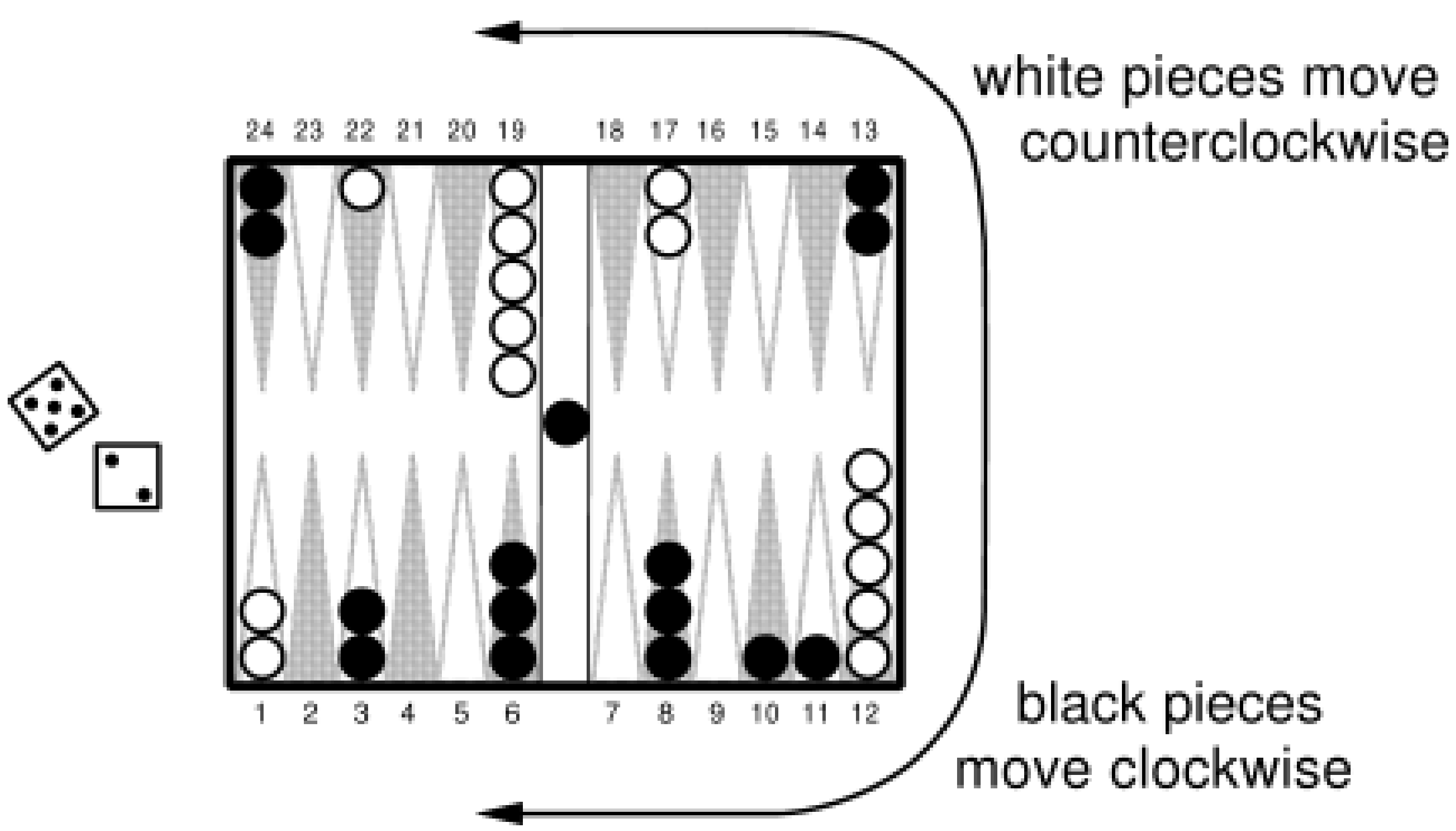
Source: https://towardsdatascience.com/cartpole-introduction-to-reinforcement-learning-ed0eb5b58288

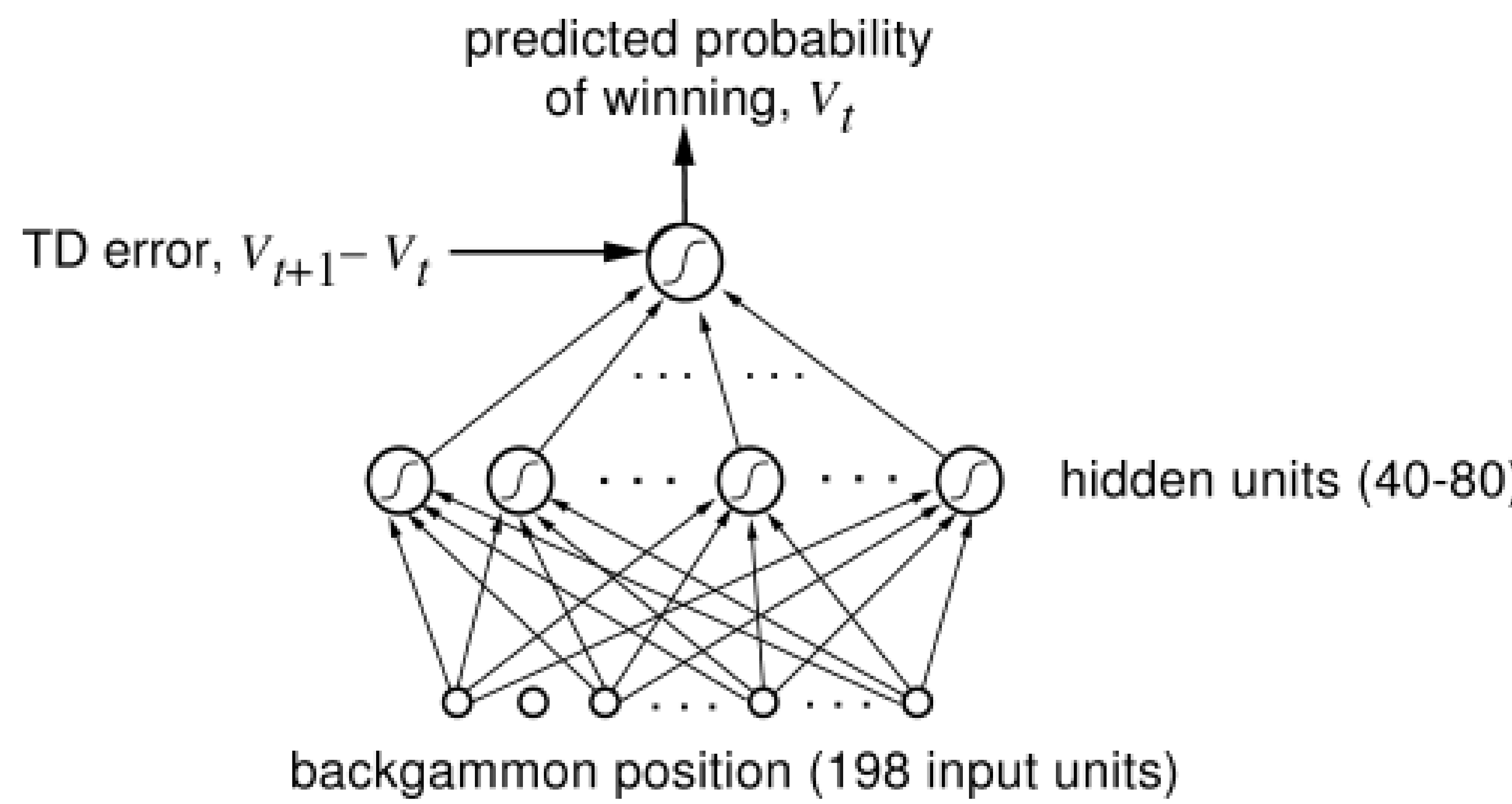
Optimal control



Board games (Backgammon, Chess, Go, etc)

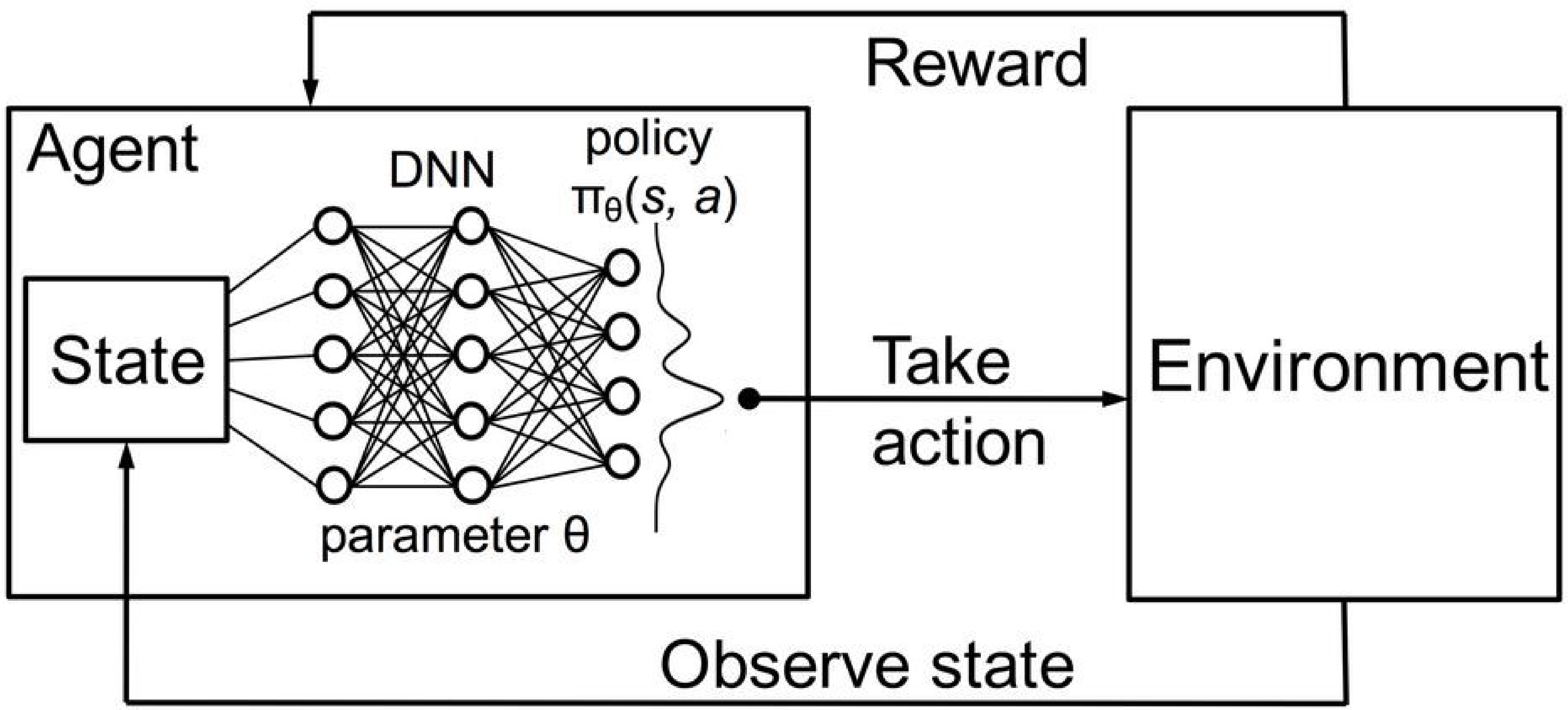
TD-Gammon (Tesauro, 1992) was one of the first AI to beat human experts at a complex game, Backgammon.





- **States**: board configurations.
- **Actions**: piece displacements.
- **Rewards**: +1 for game won, -1 for game lost, 0 otherwise. **sparse rewards**

Deep Reinforcement Learning (DRL)



- Classical tabular RL was limited to toy problems, with few states and actions.
- It is only when coupled with **deep neural networks** that interesting applications of RL became possible.
- Deepmind (now Google) started the deep RL hype in 2013 by learning to solve 50+ Atari games with a CNN.

Atari games

- **States**:

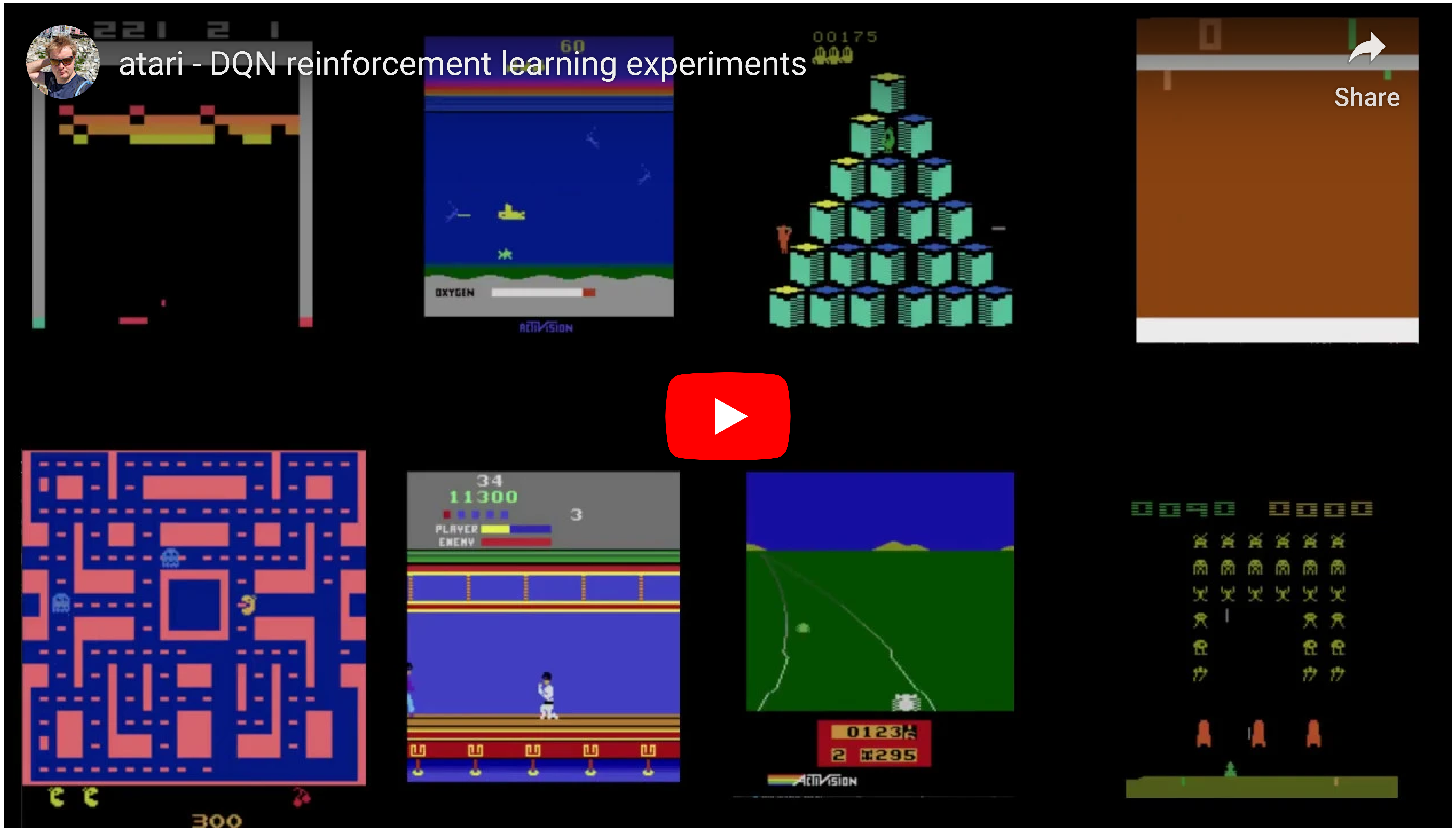
pixel frames.

- **Actions**:

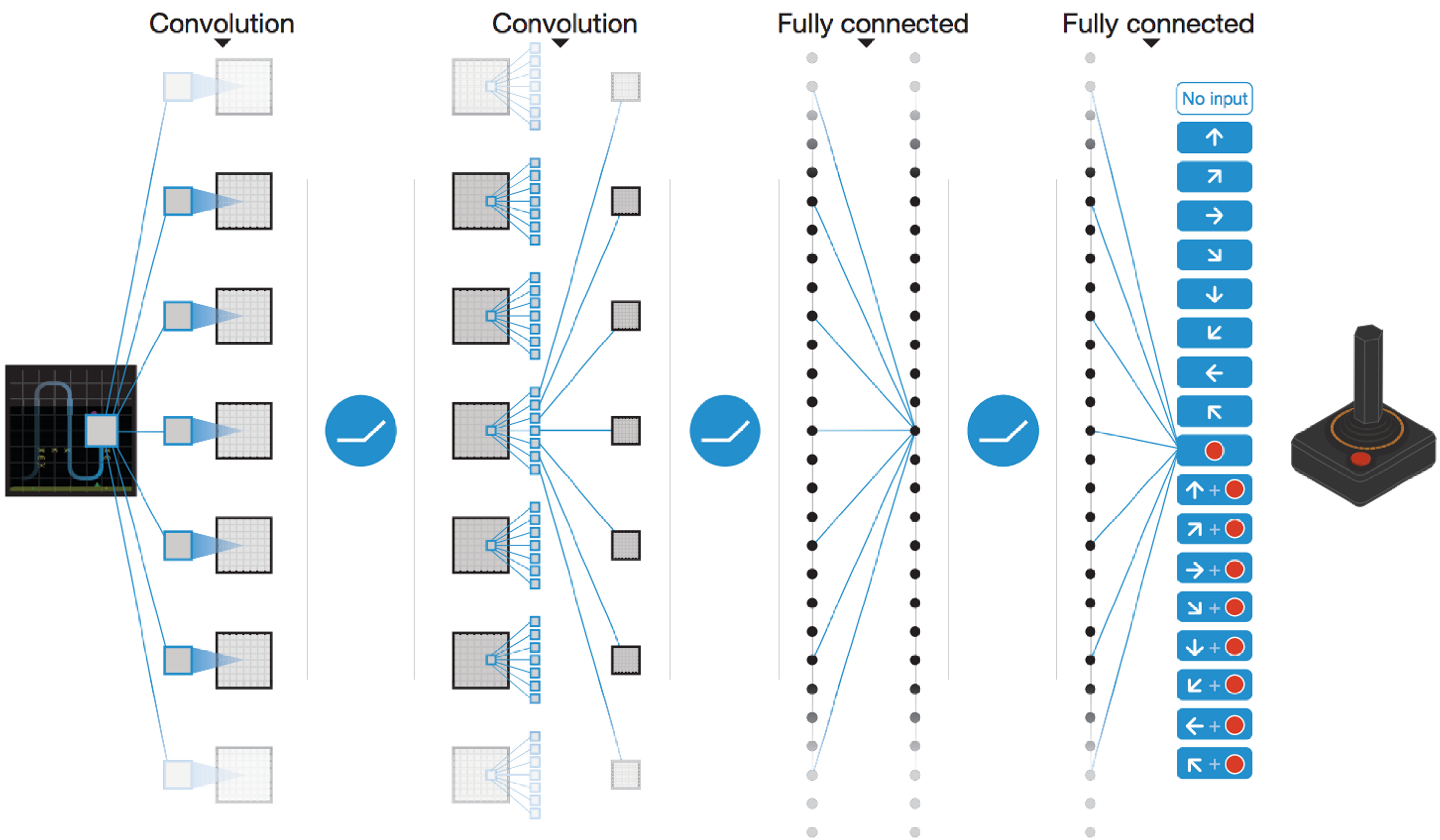
button presses.

- **Rewards**:

score increases.



Atari games



Mnih et al. (2013) Playing Atari with Deep Reinforcement Learning. NIPS. http://arxiv.org/abs/1312.5602

Simulated cars

- **States**:

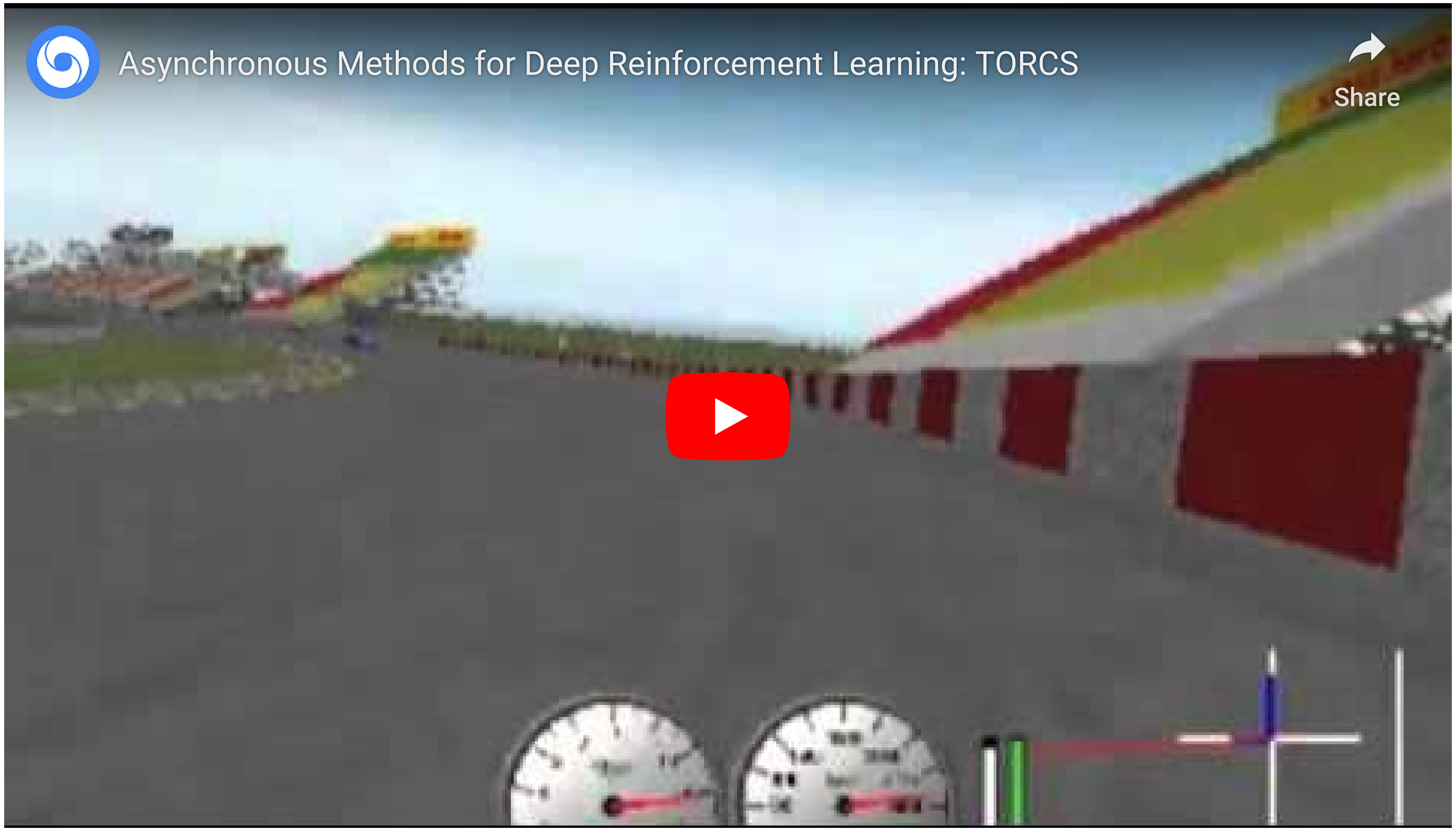
pixel frames.

- **Actions**:

direction, speed.

- **Rewards**:

linear velocity (+), crashes (-)



Parkour

- **States**:

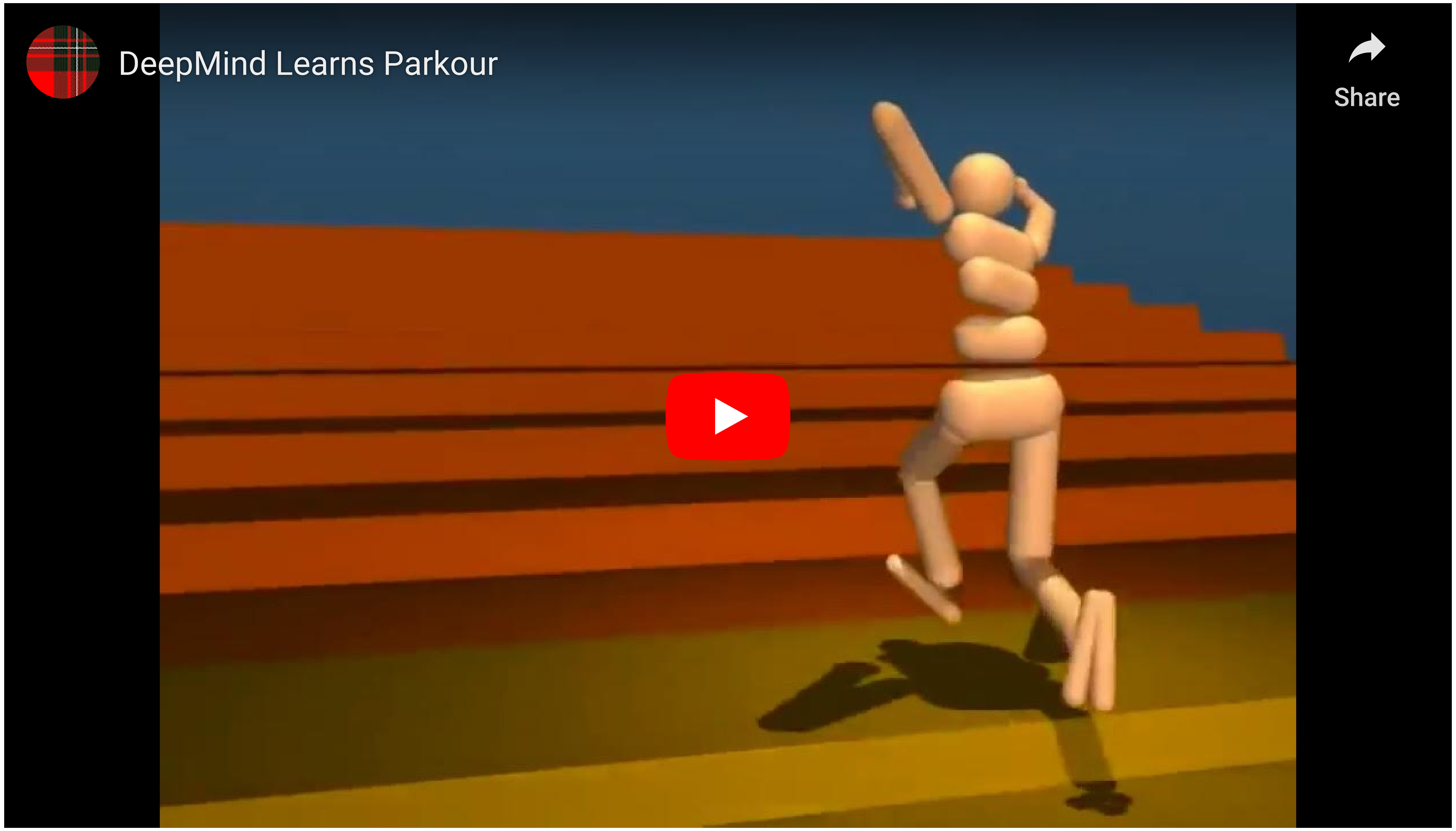
joint positions.

- **Actions**:

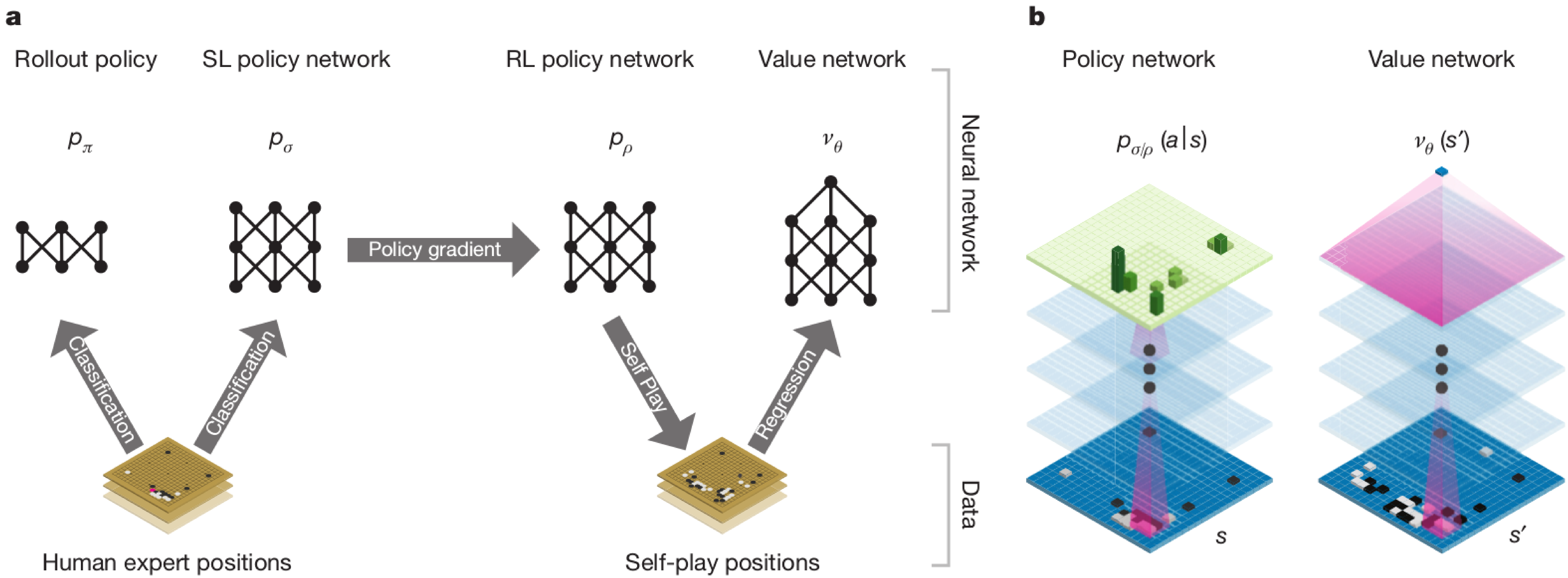
joint displacements.

- **Rewards**:

linear velocity (+), crashes (-)



AlphaGo

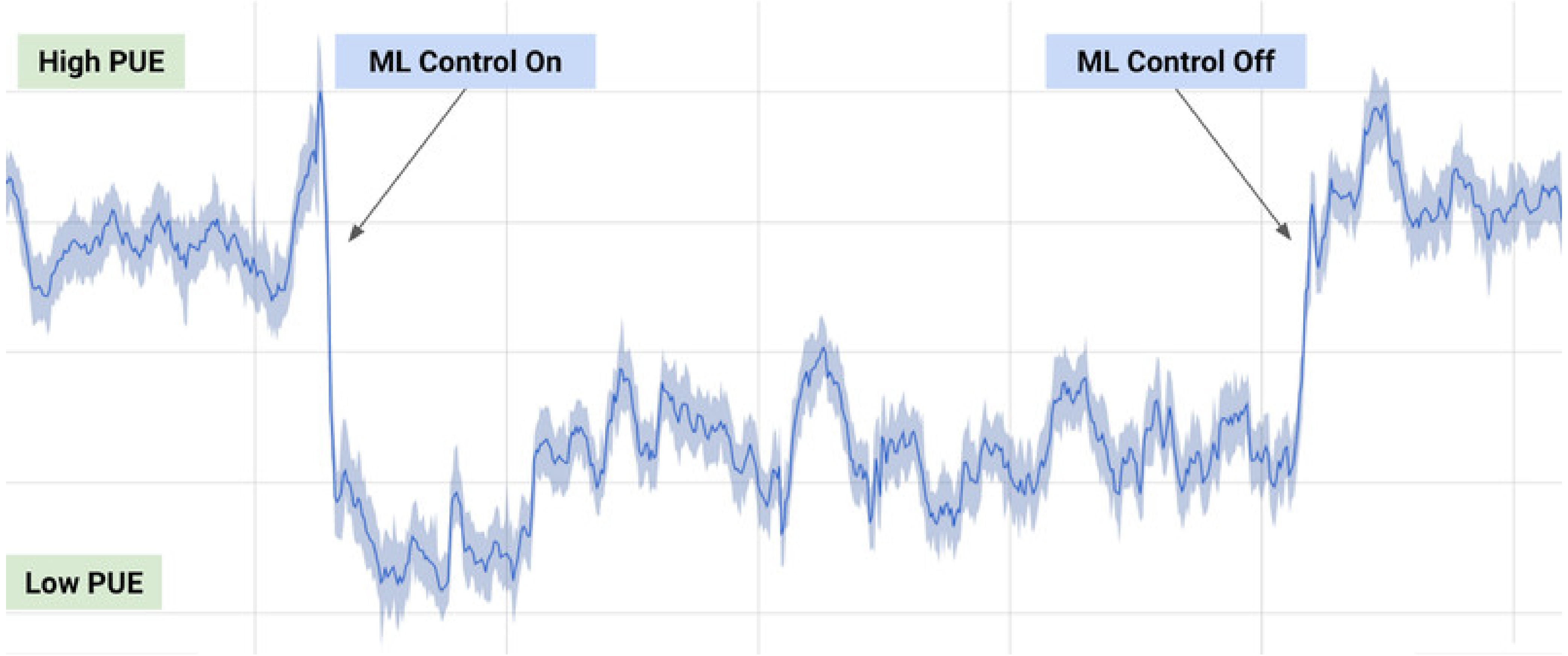


- AlphaGo was able to beat Lee Sedol in 2016, 19 times World champion.
- It relies on human knowledge to **bootstrap** a RL agent (supervised learning).
- The RL agent discovers new strategies by using self-play: during the games against Lee Sedol, it was able to use **novel** moves which were never played before and surprised its opponent.
- Training took several weeks on 1202 CPUs and 176 GPUs.

Silver et al. (2016) Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587):484–489, arXiv:1712.01815

AlphaGo

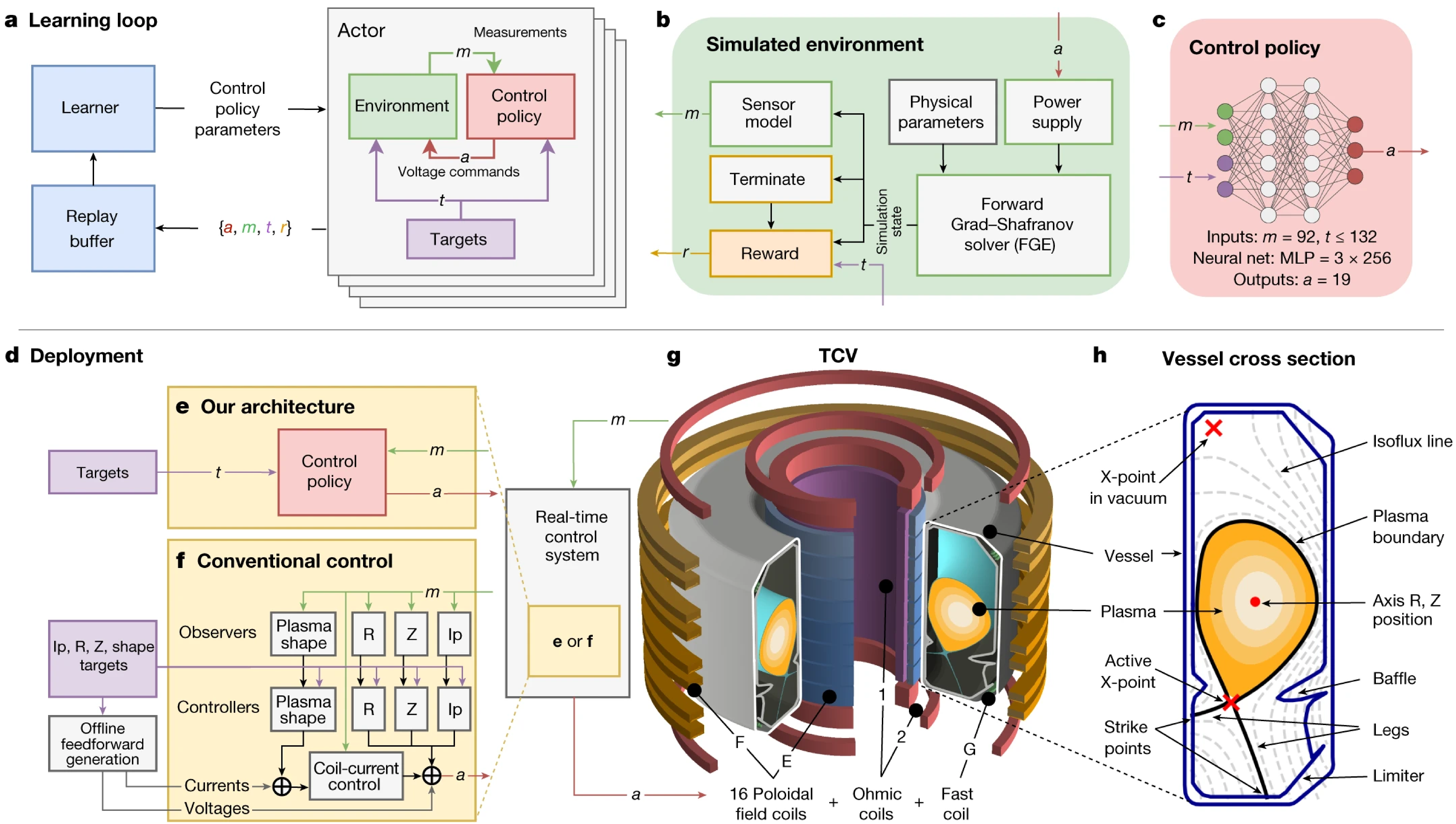
Process control



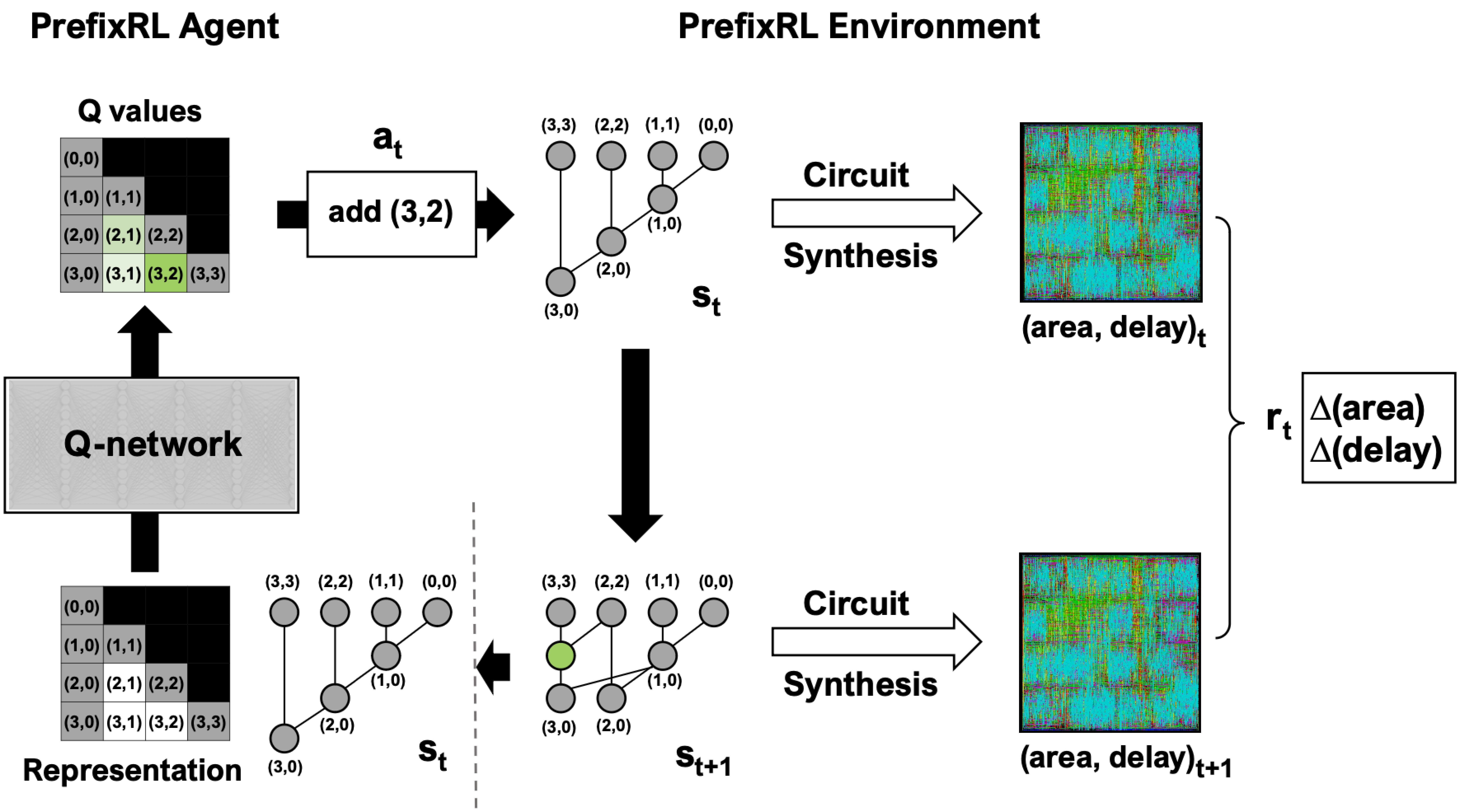
Source: https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40/

- 40% reduction of energy consumption when using deep RL to control the cooling of Google's datacenters.
- **States:** sensors (temperature, pump speeds).
- **Actions:** 120 output variables (fans, windows).
- **Rewards:** decrease in energy consumption

Magnetic control of tokamak plasmas



Chip design



Roy et al. (2022) PrefixRL: Optimization of Parallel Prefix Circuits using Deep Reinforcement Learning. doi:10.1109/DAC18074.2021.9586094

Real robotics

- **States:**

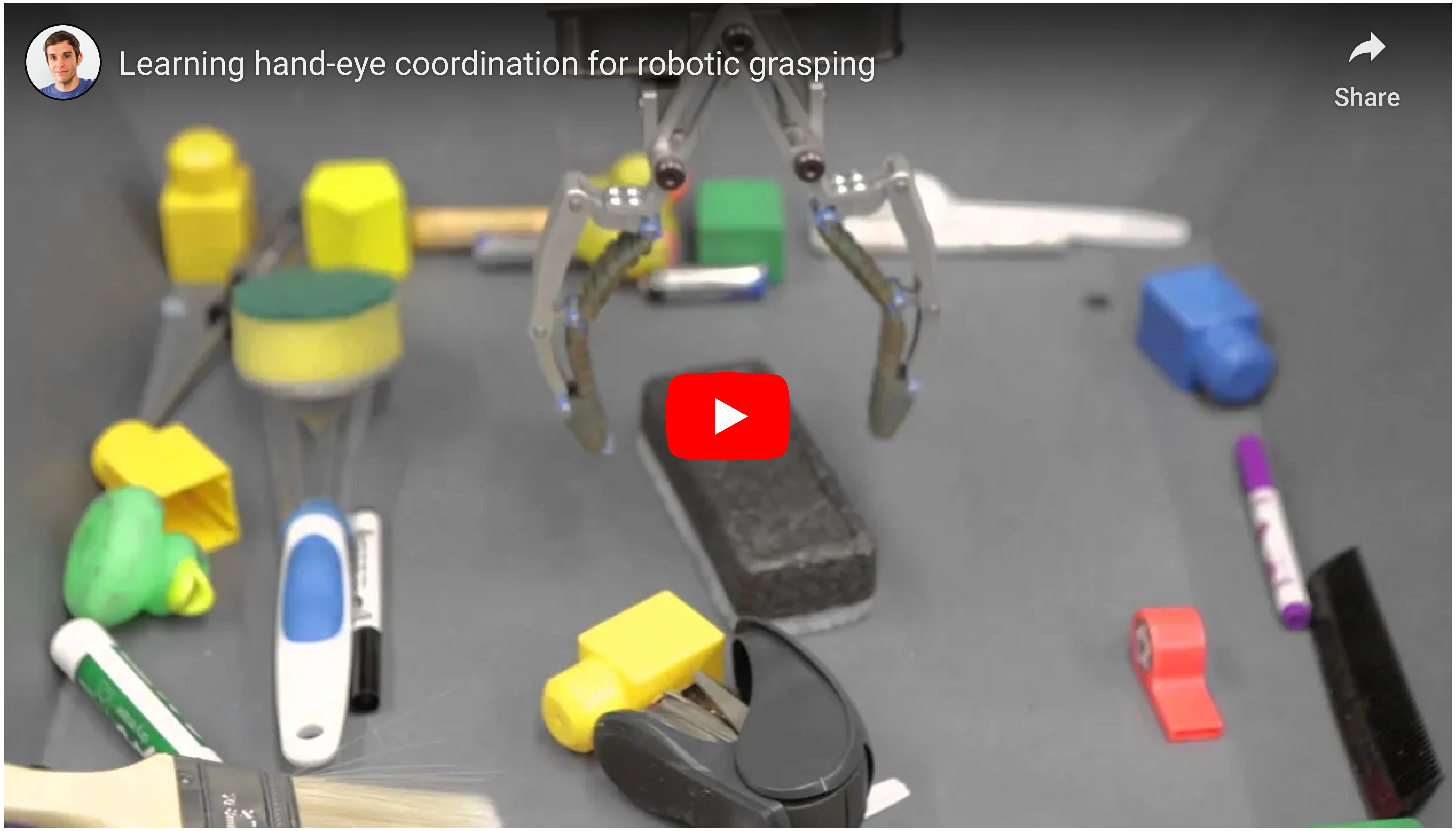
pixel frames.

- **Actions:**

joint movements.

- **Rewards:**

successful grasping.



Learning dexterity

- **States:**

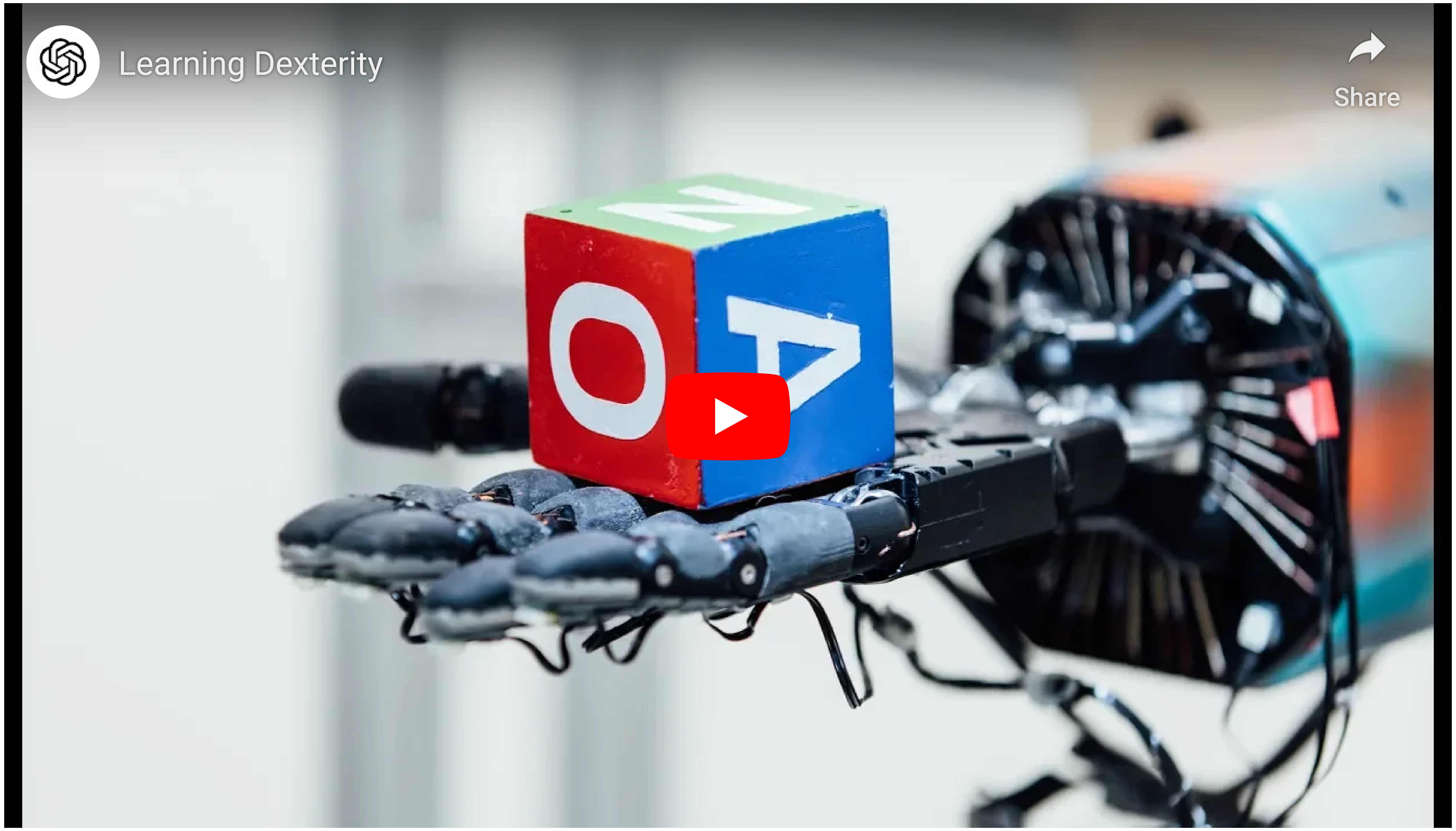
pixel frames, joint position.

- **Actions:**

joint movements.

- **Rewards:**

shape obtained.



Autonomous driving

- **States:**

pixel frames.

- **Actions:**

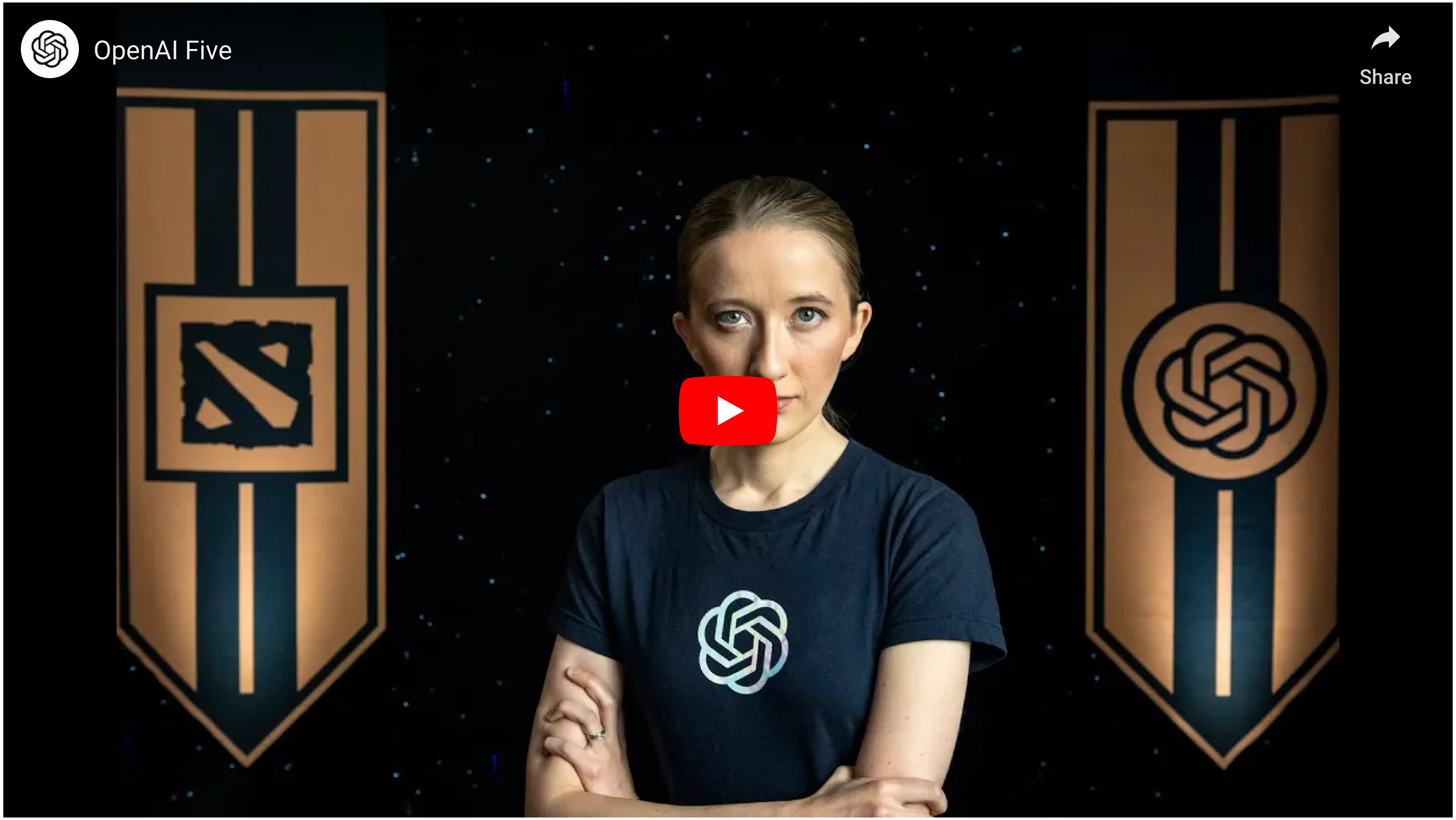
direction, speed.

- **Rewards:**

time before humans take control.

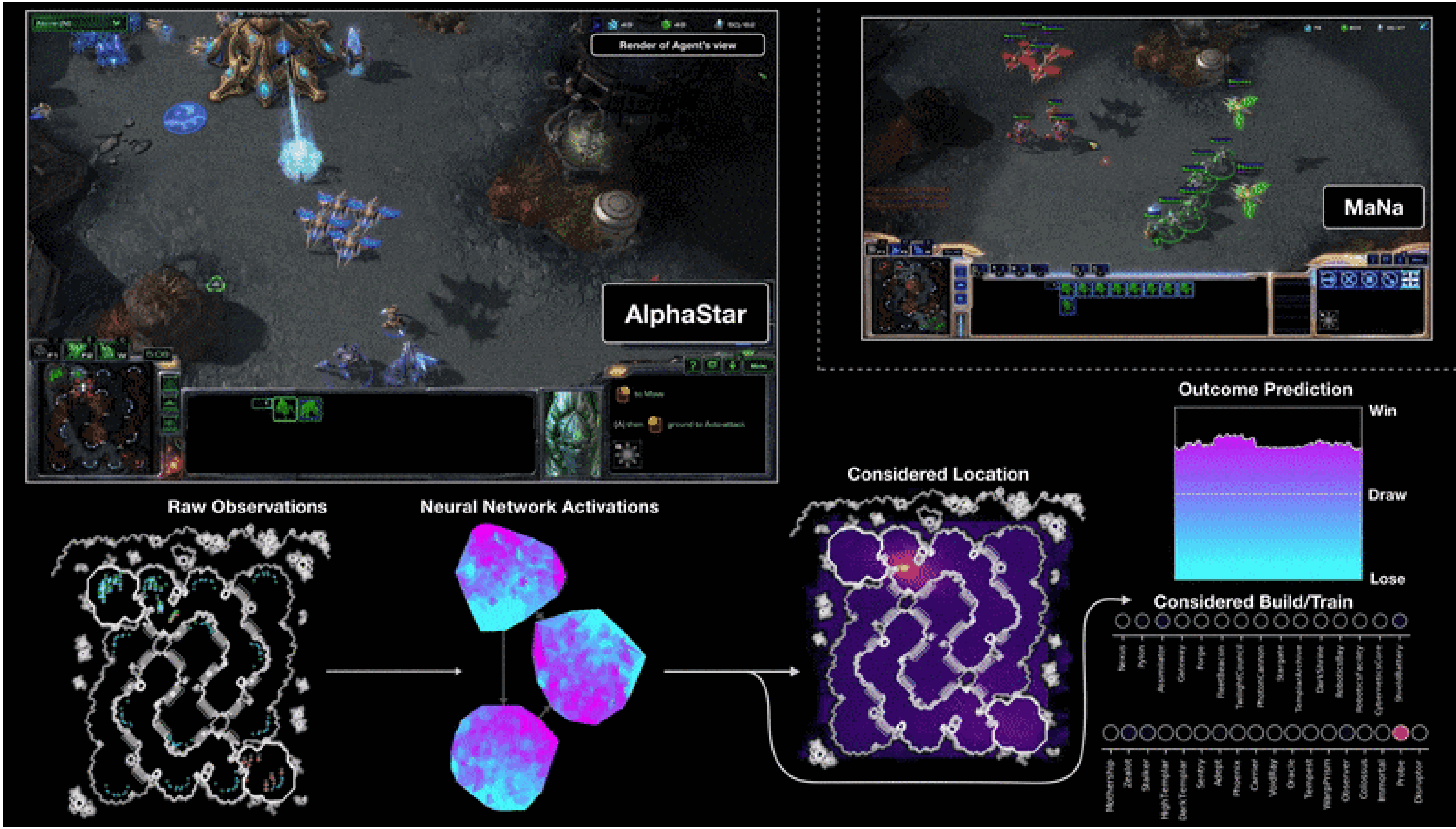
https://wayve.ai/blog/learning-to-drive-in-a-day-with-reinforcement-learning

Dota2 (OpenAI)



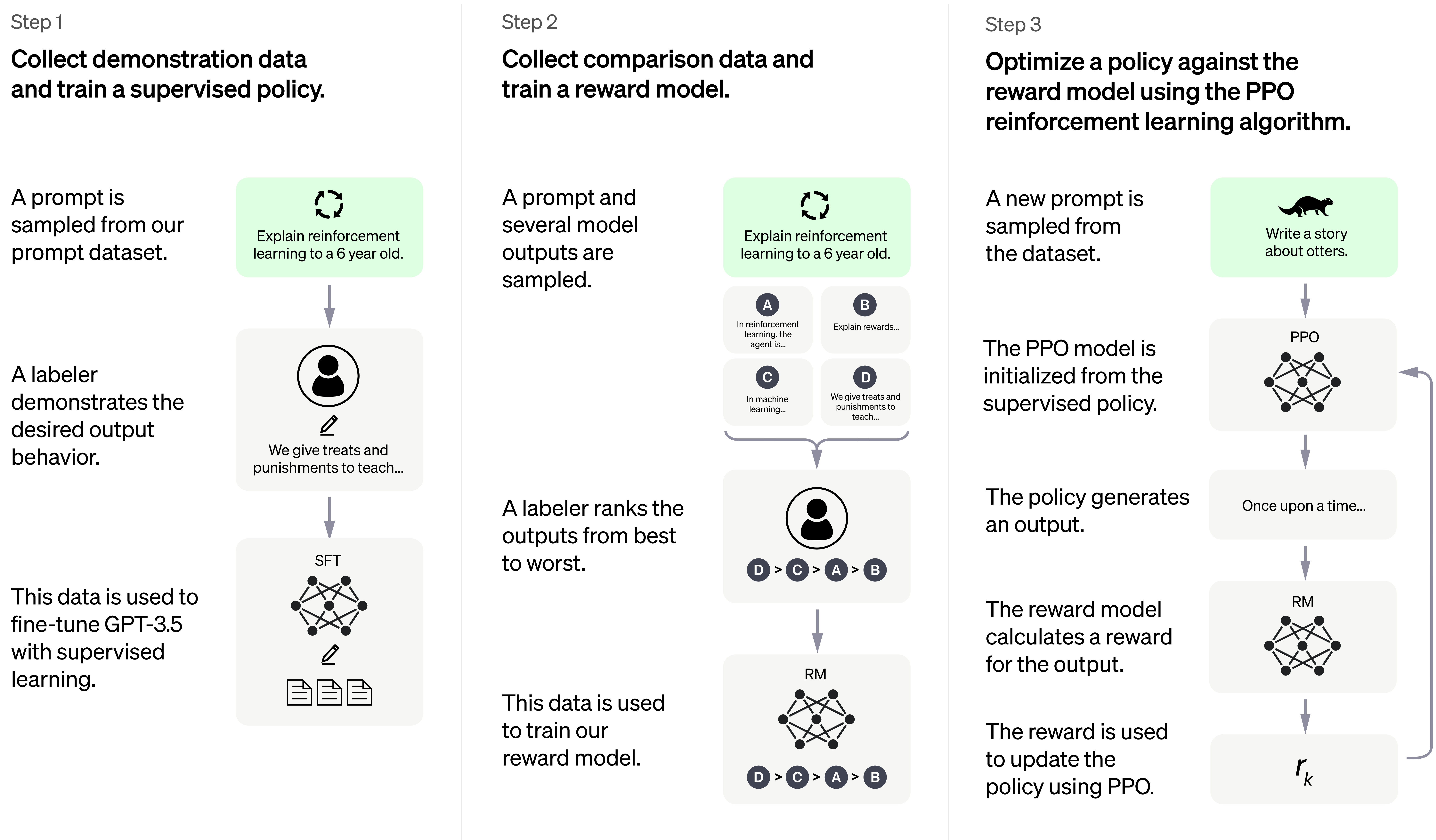
- 128,000 CPU cores and 256 Nvidia P100 GPUs on Google Cloud for 10 months ($25,000 per day)…

Starcraft II (AlphaStar)



Source: https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii

ChatGPT


Step 1
Collect demonstration data and train a supervised policy.
A prompt is sampled from our prompt dataset.
Explain reinforcement learning to a 6 year old.
A labeler demonstrates the desired output behavior.
We give treats and punishments to teach...
This data is used to fine-tune GPT-3.5 with supervised learning.
SFT
Step 2
Collect comparison data and train a reward model.
A prompt and several model outputs are sampled.
Explain reinforcement learning to a 6 year old.
A In reinforcement learning, the agent is...
B Explain rewards...
C In machine learning...
D We give treats and punishments to teach...
A labeler ranks the outputs from best to worst.
D > C > A > B
This data is used to train our reward model.
RM
D > C > A > B
Step 3
Optimize a policy against the reward model using the PPO reinforcement learning algorithm.
A new prompt is sampled from the dataset.
Write a story about otters.
The PPO model is initialized from the supervised policy.
PPO
The policy generates an output.
Once upon a time...
The reward model calculates a reward for the output.
RM
The reward is used to update the policy using PPO.
r_k

Take home messages

- Deep RL is gaining a lot of importance in AI research.
  - Lots of applications in control: video games, robotics, industrial applications…
  - It may be AI's best shot at producing intelligent behavior, as it does not rely on annotated data.
- A lot of problems have to be solved before becoming as mainstream as deep learning.
  - Sample complexity is often prohibitive.
  - Energy consumption and computing power simply crazy (AlphaGo: 1 MW, Dota2: 800 petaflop/s-days)
  - The correct reward function is hard to design, ethical aspects. (*inverse RL*)
  - Hard to incorporate expert knowledge. (*model-based RL*)
  - Learns single tasks, does not generalize (*hierarchical RL*, *meta-learning*)

Plan of the course

1. **Introduction**
    1. Applications
    2. Crash course in statistics
2. **Basic RL**
    1. Bandits
    2. Markov Decision Process
    3. Dynamic programming
    4. Monte-Carlo control
    5. Temporal difference, Eligibility traces
    6. Function approximation
    7. Deep learning
3. **Model-free RL**
    1. Deep Q-networks
    2. Beyond DQN
    3. Policy gradient, REINFORCE
    4. Advantage Actor-critic (A3C)
    5. Deterministic policy gradient (DDPG)
    6. Natural gradients (TRPO, PPO)
    7. Maximum Entropy RL (SAC)
4. **Model-based RL**
    1. Principle, Dyna-Q, MPC
    2. Learned World models
    3. AlphaGo
    4. Successor representations
5. **Outlook**
    1. Hierarchical RL
    2. Inverse RL
    3. Meta RL
    4. Offline RL

Suggested reading

- Sutton and Barto (1998, 2017). Reinforcement Learning: An Introduction. MIT Press.

http://incompleteideas.net/sutton/book/the-book.html

- Szepesvari (2010). Algorithms for Reinforcement Learning. Morgan and Claypool.

http://www.ualberta.ca/~szepesva/papers/RLAlgsInMDPs.pdf

- CS294 course of Sergey Levine at Berkeley.

http://rll.berkeley.edu/deeprlcourse/

- Reinforcement Learning course by David Silver at UCL.

http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html

References

Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., et al. (2022). Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature* 602, 414–419. doi:10.1038/s41586-021-04301-9.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., et al. (2013). Playing Atari with Deep Reinforcement Learning. http://arxiv.org/abs/1312.5602.

Roy, R., Raiman, J., Kant, N., Elkin, I., Kirby, R., Siu, M., et al. (2022). PrefixRL: Optimization of Parallel Prefix Circuits using Deep Reinforcement Learning. doi:10.1109/DAC18074.2021.9586094.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., et al. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 484–489. doi:10.1038/nature16961.

Sutton, R. S., and Barto, A. G. (1998). *Reinforcement Learning: An introduction*. Cambridge, MA: MIT press.

Sutton, R. S., and Barto, A. G. (2017). *Reinforcement Learning: An Introduction*. 2nd ed. Cambridge, MA: MIT Press http://incompleteideas.net/book/the-book-2nd.html.