Stationary Probability Distribution: This means that the pattern of rewards for each arm doesn't change over time

COMP47590: Advanced Machine Learning

Reinforcement Learning - Introduction

Dr. Brian Mac Namee

k-ARMED BANDIT PROBLEM

One-armed bandit slot machine pays out a prize at a fixed probability

***k*-armed bandit** slot machine pays out a prize at different fixed probabilities for each arm

Multi-armed Bandit Problem

You are faced repeatedly with a choice among k different arms on the sot machine

After each choice you receive a numerical reward (0 or some positive value) chosen from a stationary probability distribution that depends on the arm you selected

Your objective is to maximize the expected total reward over some time period, for example, over 1000 arm pulls

Stationary Probability Distribution: This means that the pattern of rewards for each arm doesn't change over time

k-armed Bandit Problem

On each of an infinite sequence of *time steps*, *t*=1, 2, 3, …, you choose an action A_t from *k* possibilities, and receive a real-valued *reward* R_t

The reward depends only on the action taken; it is identically, independently distributed (i.i.d.)

q*(a) =expected reward for action a = average reward i would recieve if i chose action a repeatedly

$$q_*(a) \doteq \mathbb{E}[R_t \mid A_t = a]$$

Identically, Independently Distributed (i.i.d.): This means that each time you pull an arm, the reward you get doesn't depend on previous pulls.

need to find the highest q(a)

k-armed Bandit Problem

We denote the estimated value of action a at time step t as $Q_t(a)$

We would like $Q_t(a)$ to be close to $q*(a)$ as possible

Sample average method is a simple way to estimate $Q_t(a)$

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t}$$

***k*-armed Bandit Problem**

The simplest action selection rule is to always select the actions with the highest estimated value

$$A_t \doteq \underset{a}{\arg\max}\, Q_t(a)$$

This is referred to as **greedy** action selection

The Exploration/Exploitation Dilemma

At* is the best option possible

If $A_t = A_t^*$ then you are **exploiting**

If $A_t \neq A_t^*$ then you are **exploring**

You can’t do both, but you need to do both

You can ’ t always stick to what you know, but you also can't always try new things. You need to balance both.

ε-Greedy Action Selection

In greedy action selection, you always exploit

In ε-greedy, you are usually greedy, but with probability ε you instead pick an action at random

This is perhaps the simplest way to balance exploration and exploitation

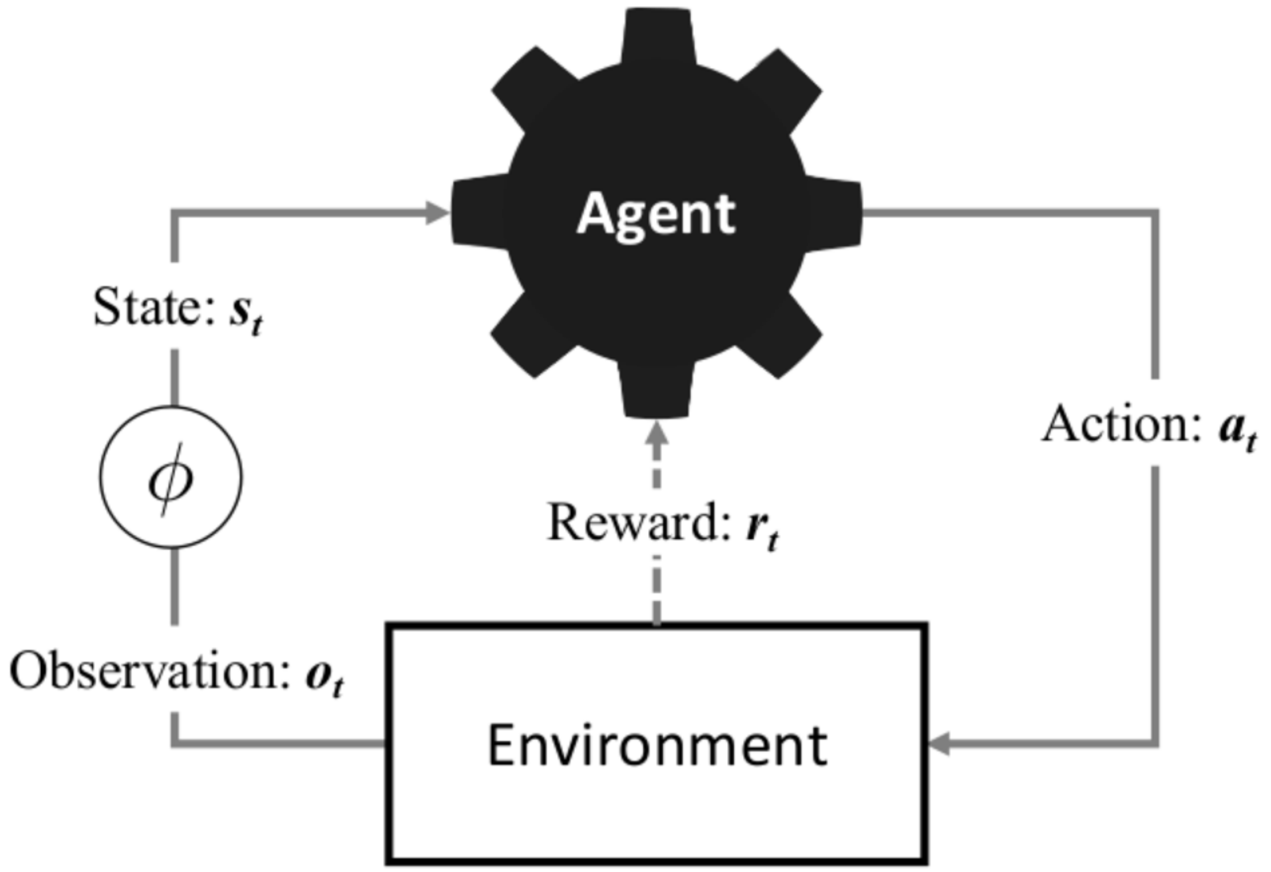
k-armed Bandit

The k-armed bandit problem is just about an example of reinforcement learning

From a reinforcement learning point of view the key components are:

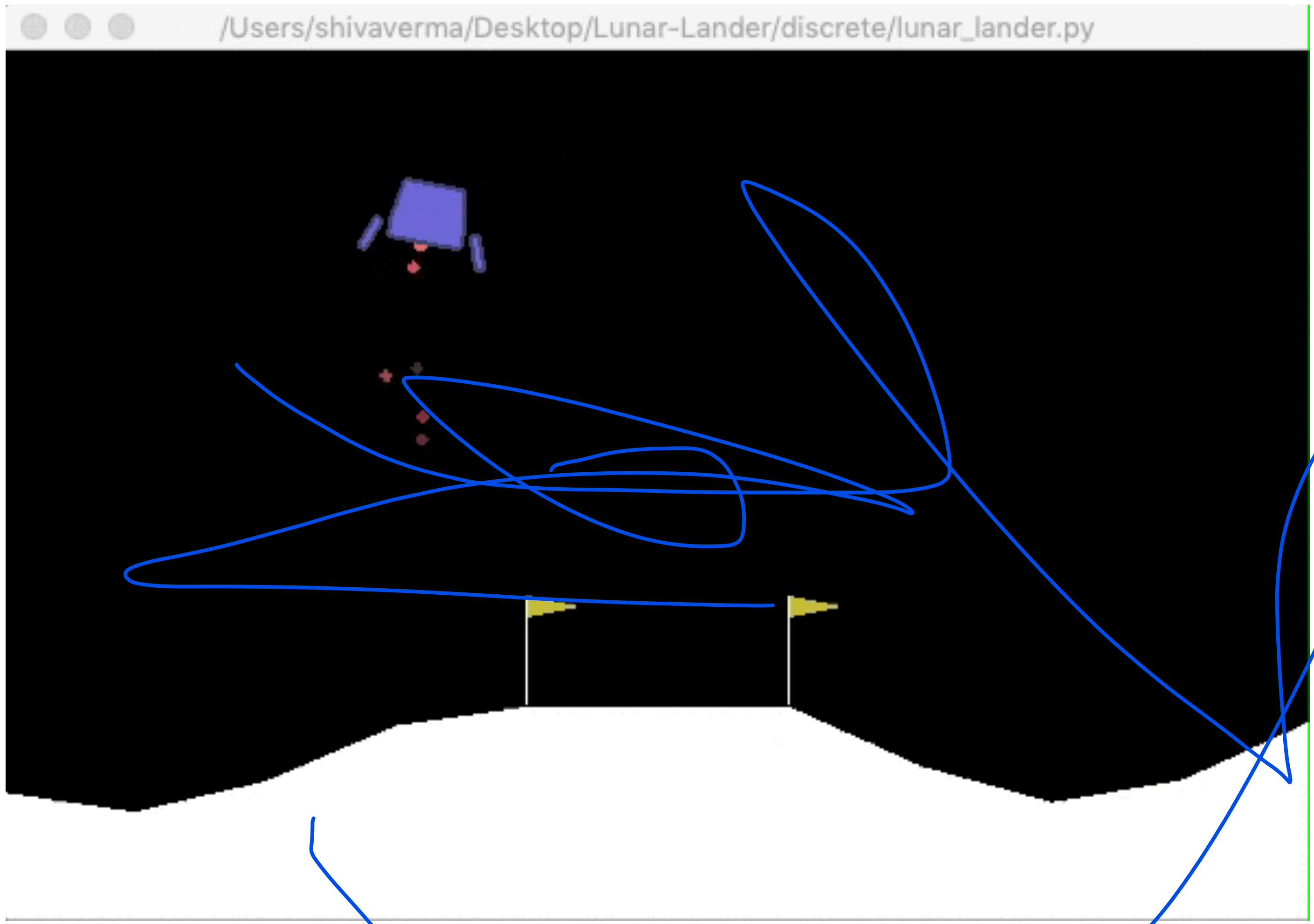
- **agent**: the arm puller
- **environment**: the bandit machine
- a **policy**: greedy or ε-greedy action selection
- a **reward signal**: R_t
- a **value function**: $Q_t(a)$
- a **state**: did you spot this?

Reinforcement Learning



Imagine you have a spaceship, and your goal is to land it safely on the moon. The spaceship has controls like thrusters to move up, down, left, and right. You need to figure out the best way to use these controls to land smoothly.

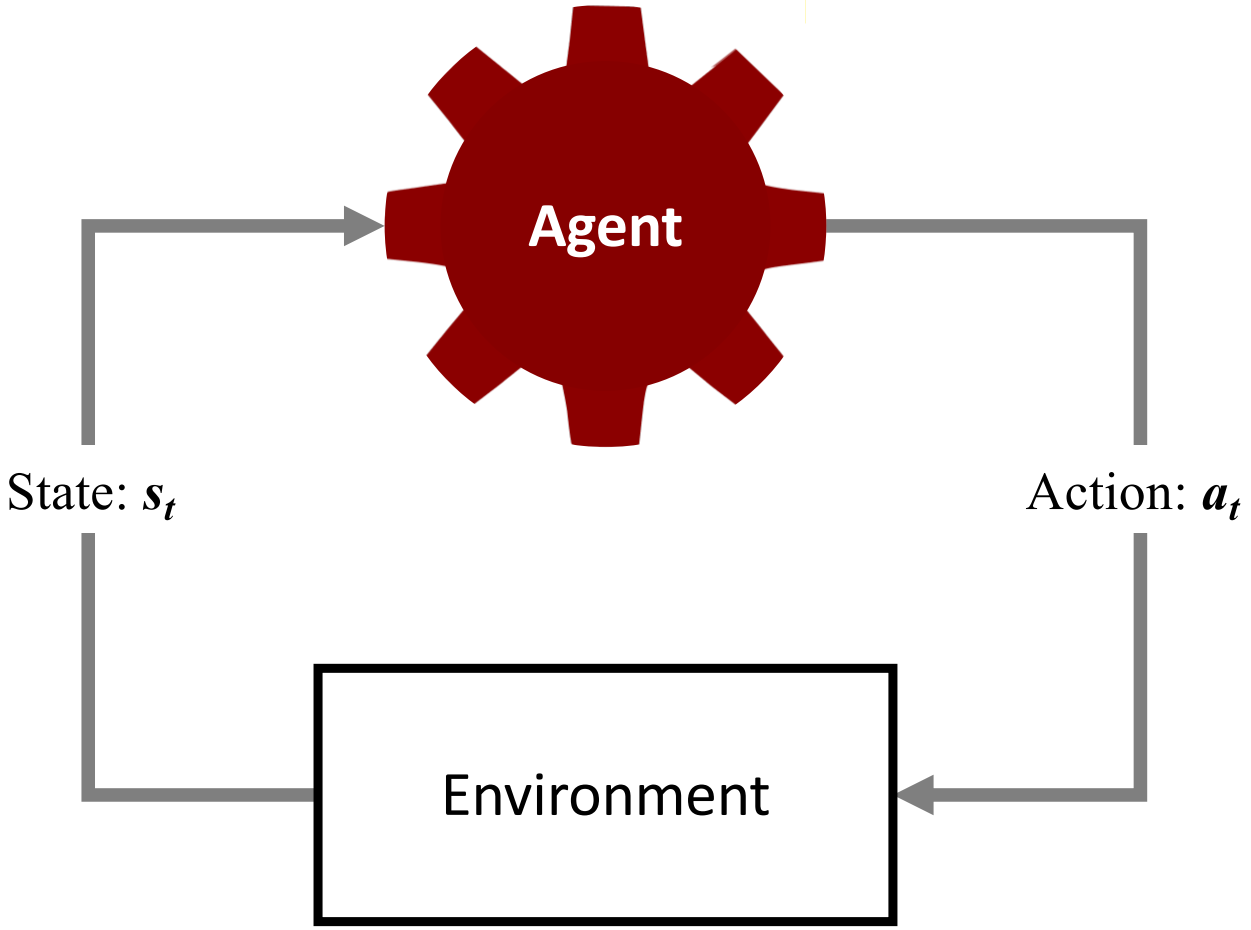
Landing On The Moon

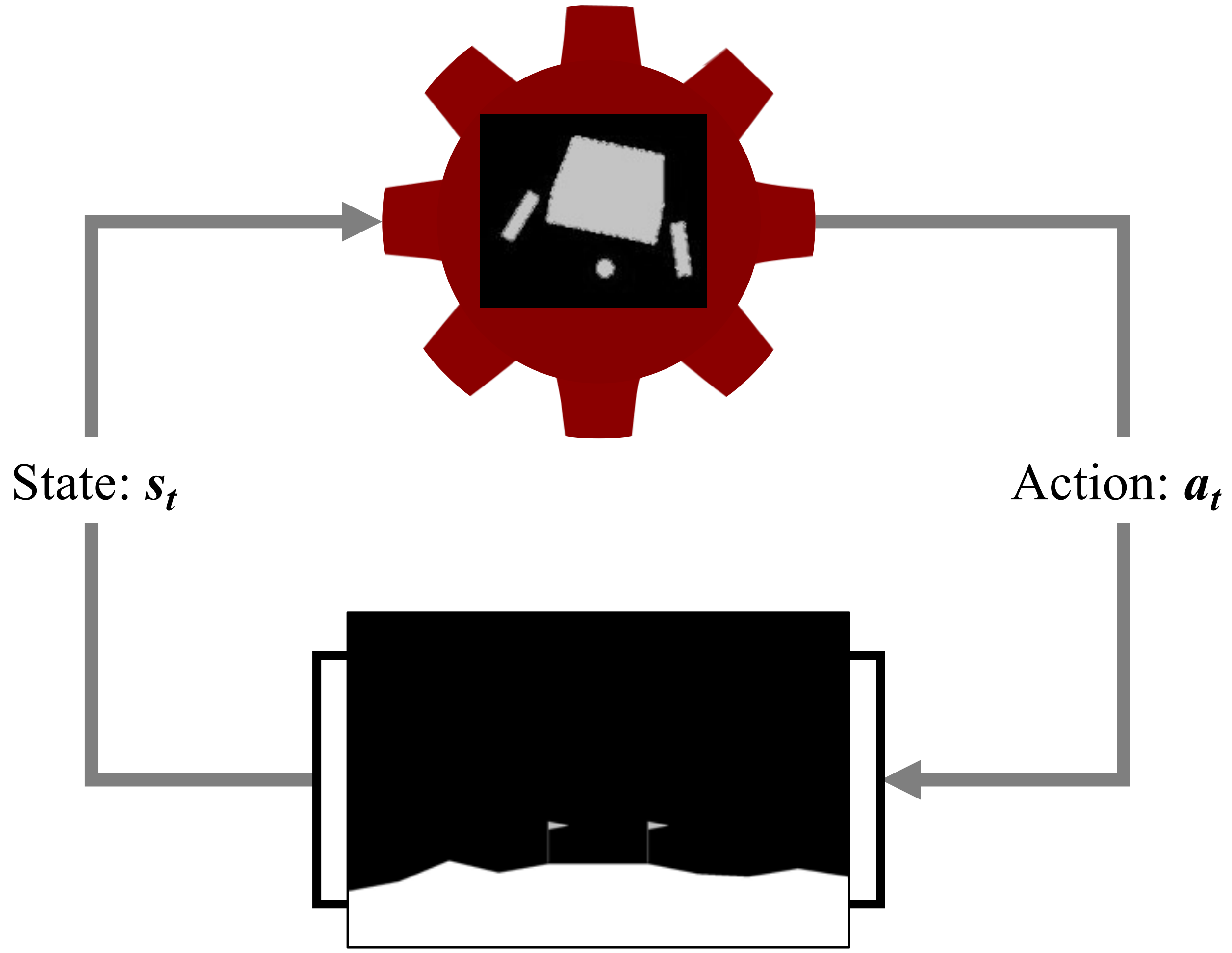

/Users/shivaverma/Desktop/Lunar-Lander/discrete/lunar_lander.py

https://gym.openai.com/envs/LunarLander-v2/

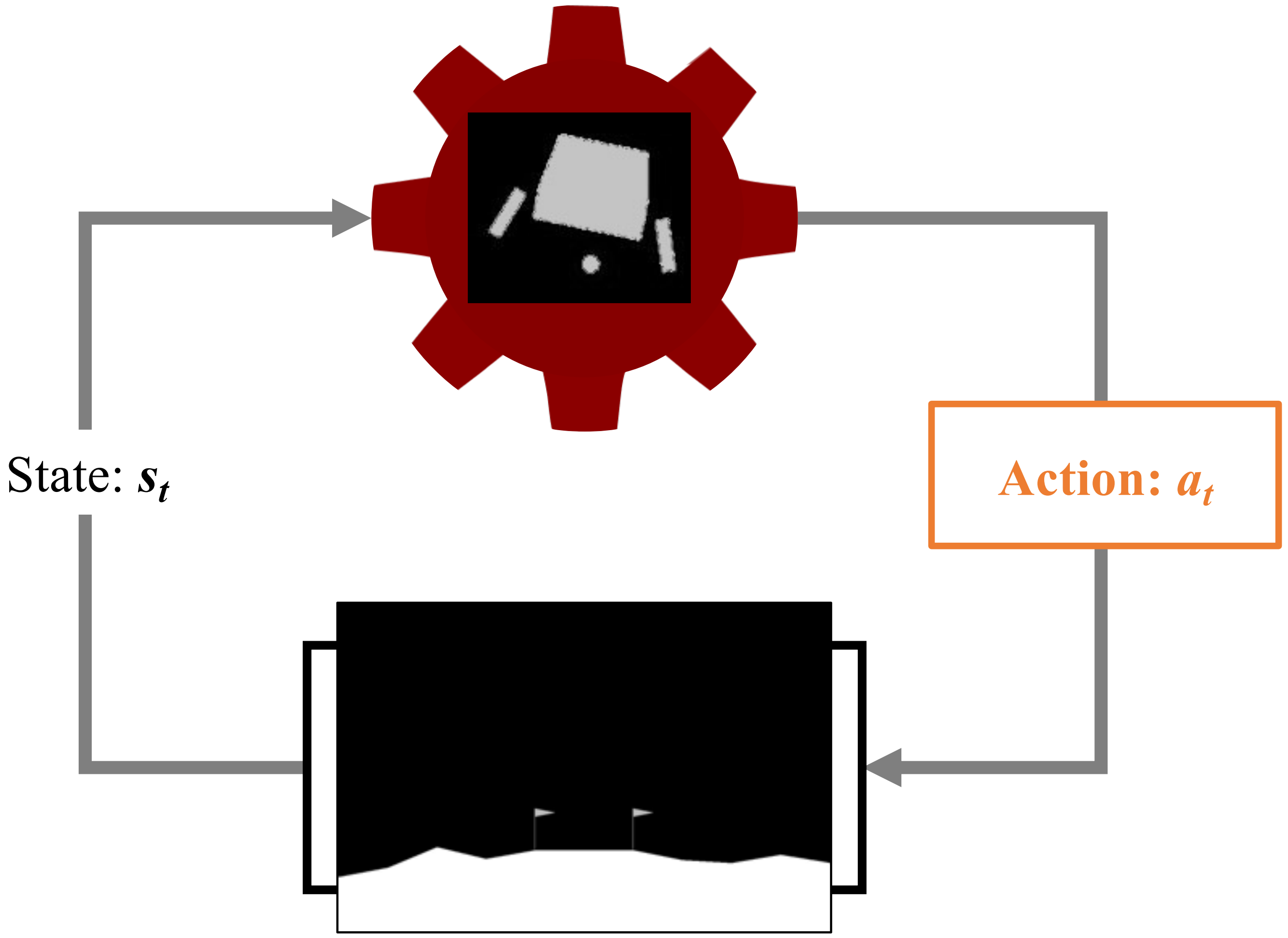
Spaceship: This is the thing you're trying to land on the moon. It’s part of the environment.

Agent: This is the "pilot" or the controller that decides what actions to take (like firing thrusters) to land the spaceship safely. The agent learns by trying different actions and seeing what works best.

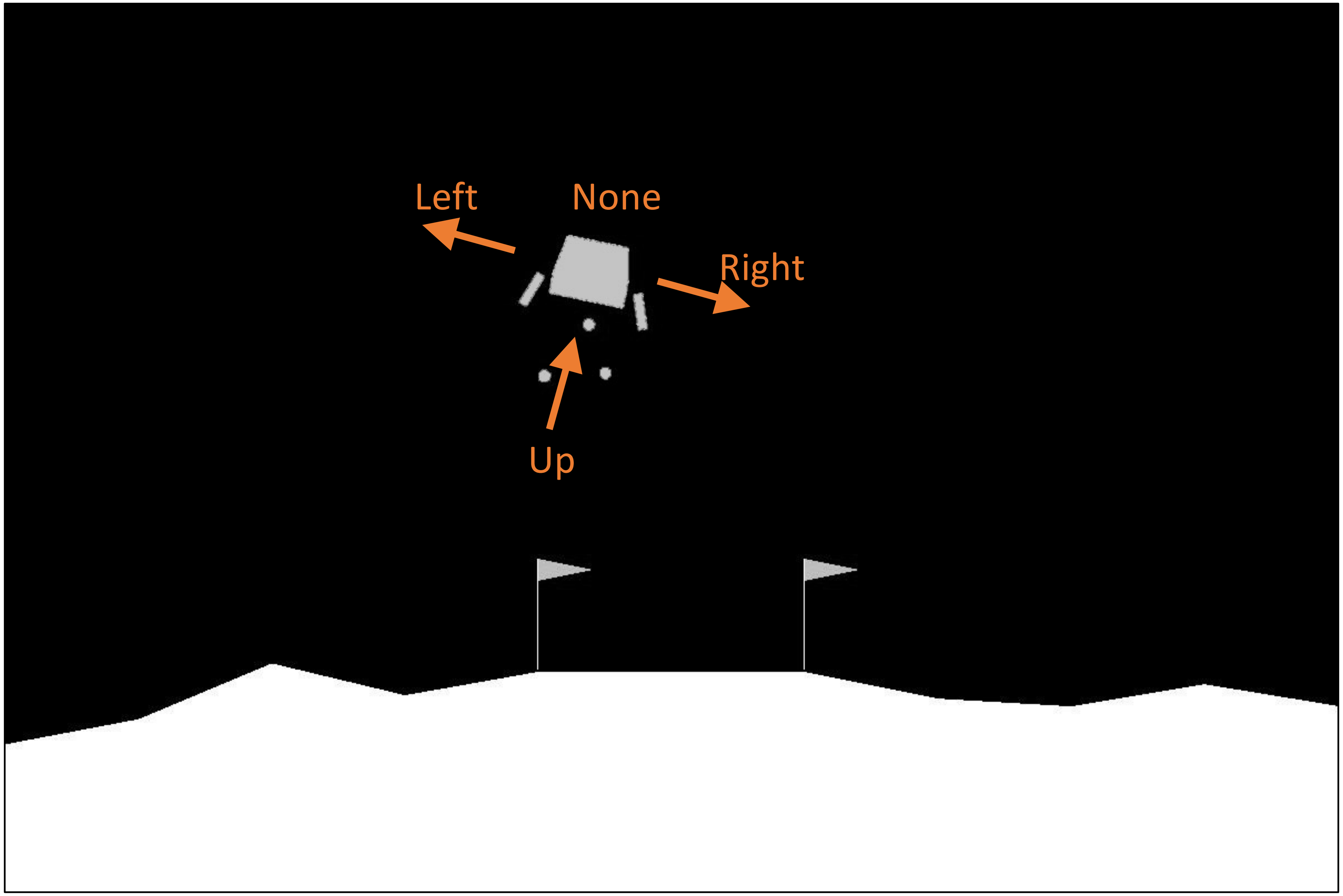


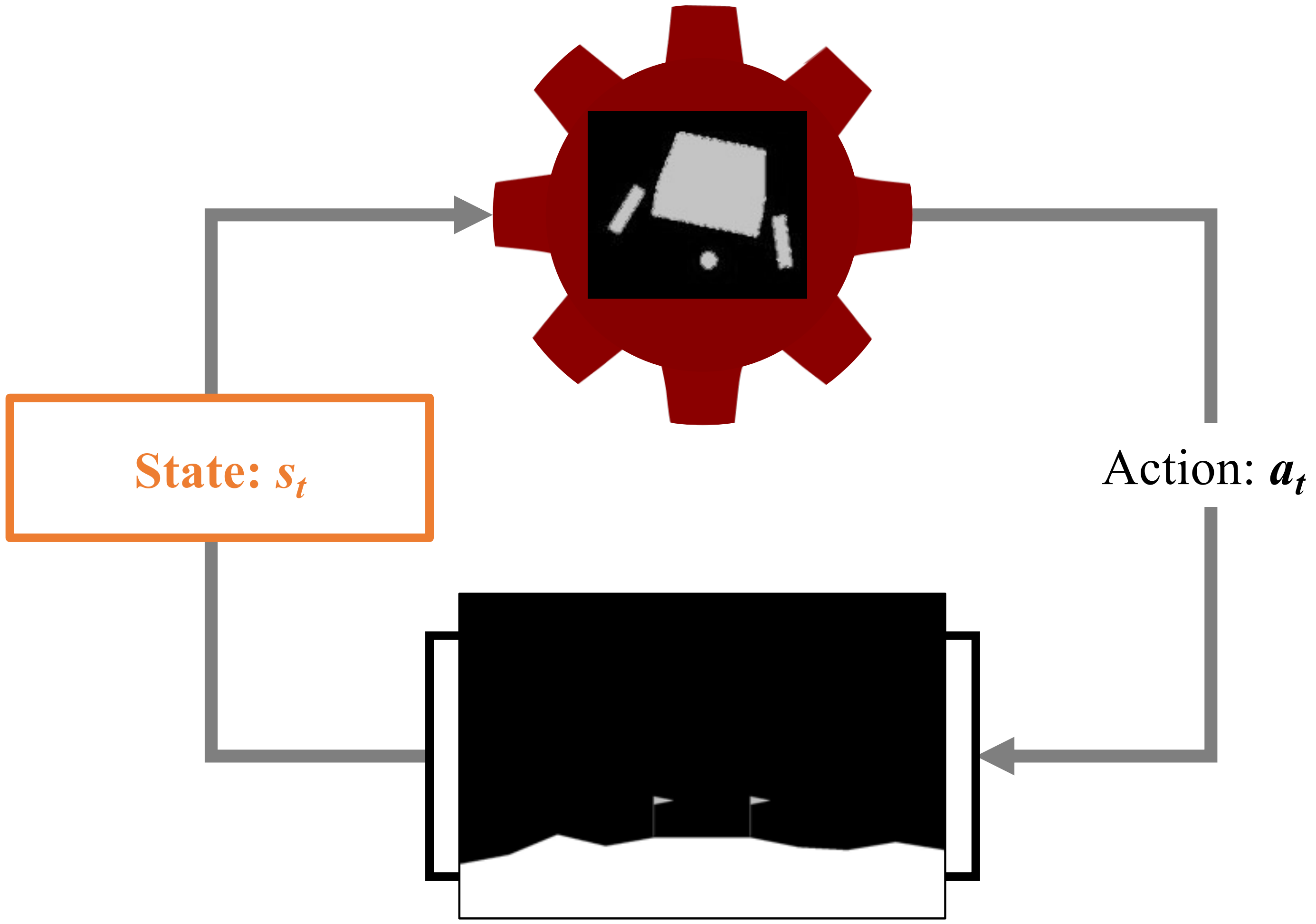


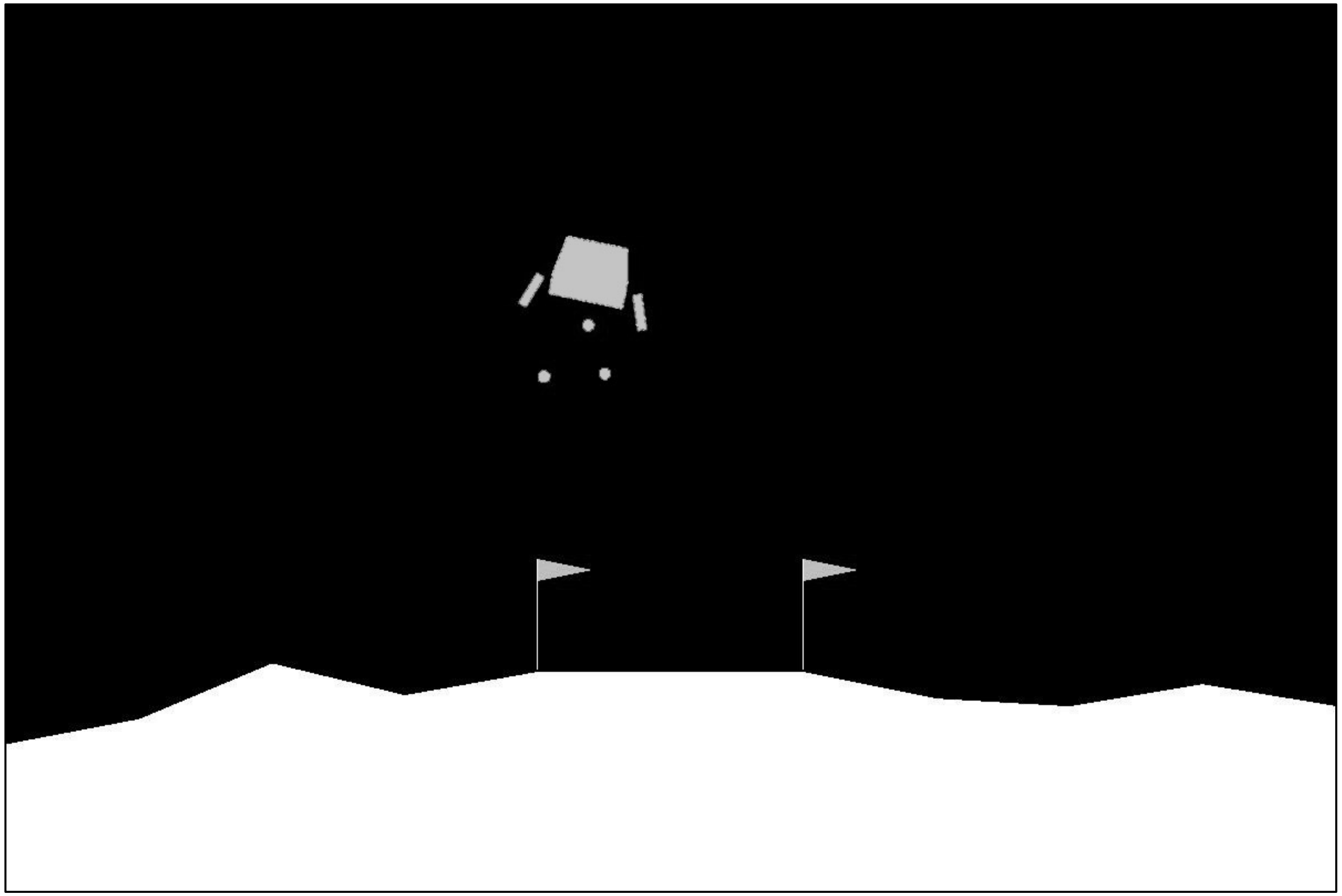
envinroment is the space and the ship

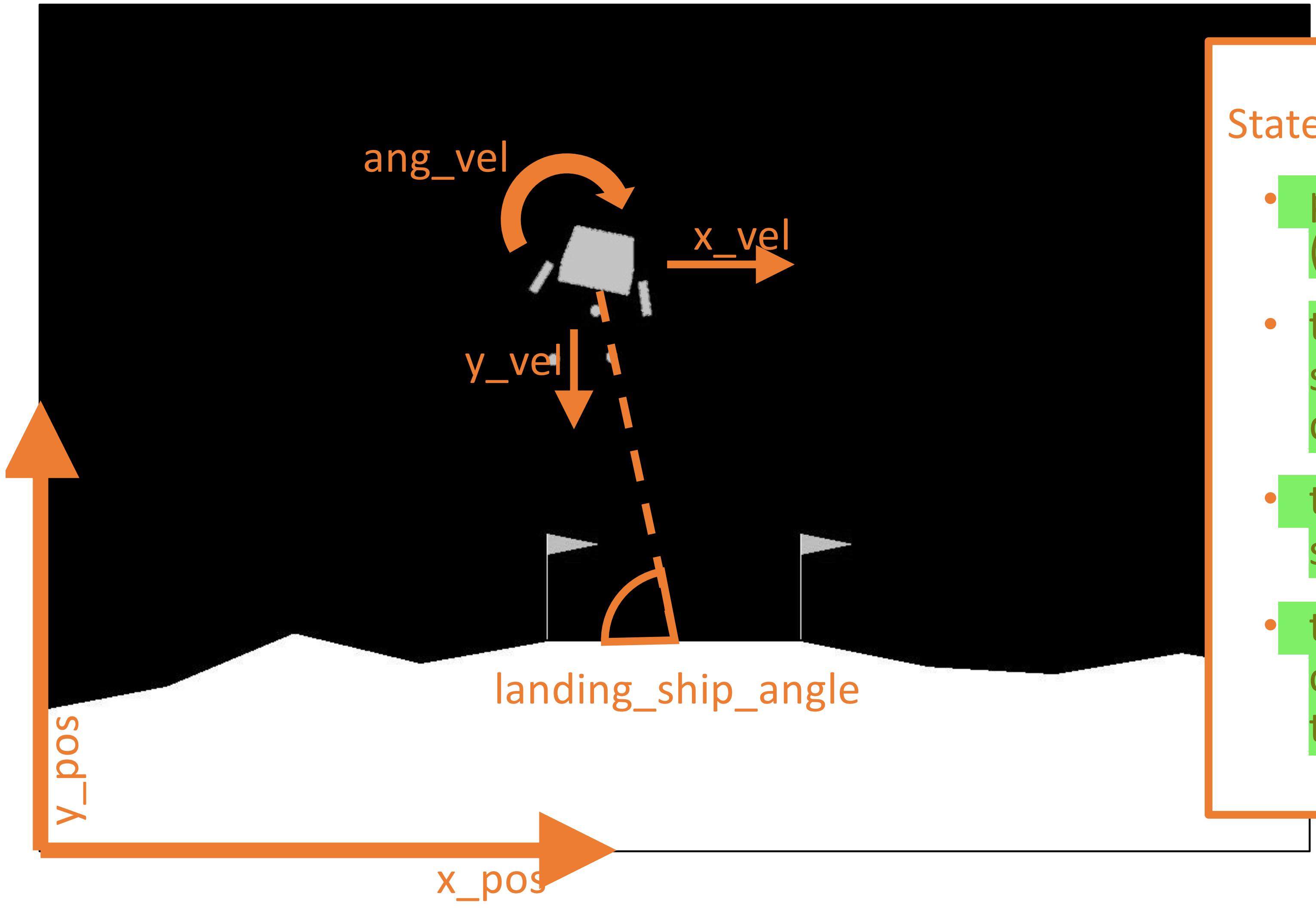


Action: These are the moves you can make, like firing the thrusters to move up, down, left, or right.


Left
None
Right
Up

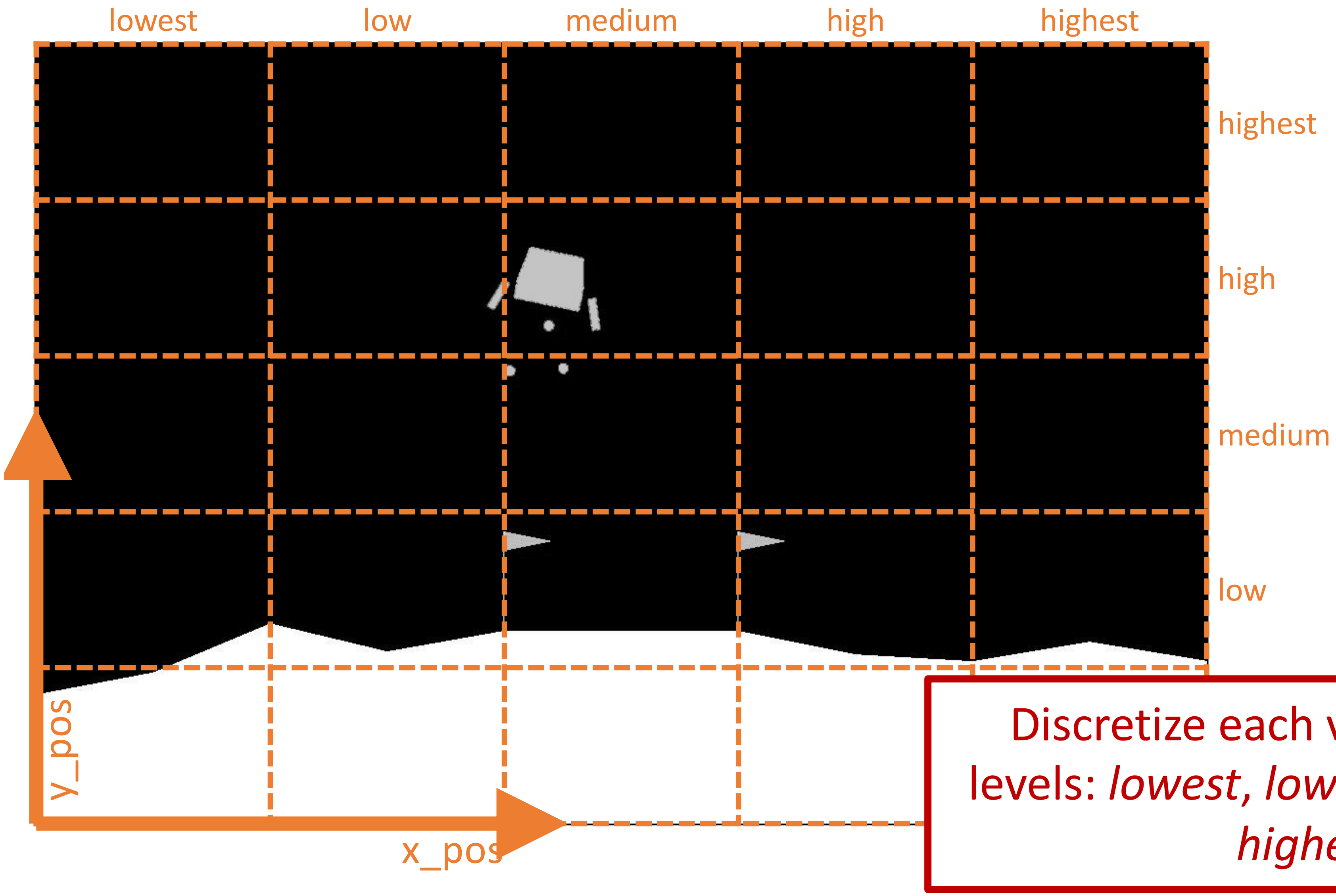

State: s_t
Action: a_t



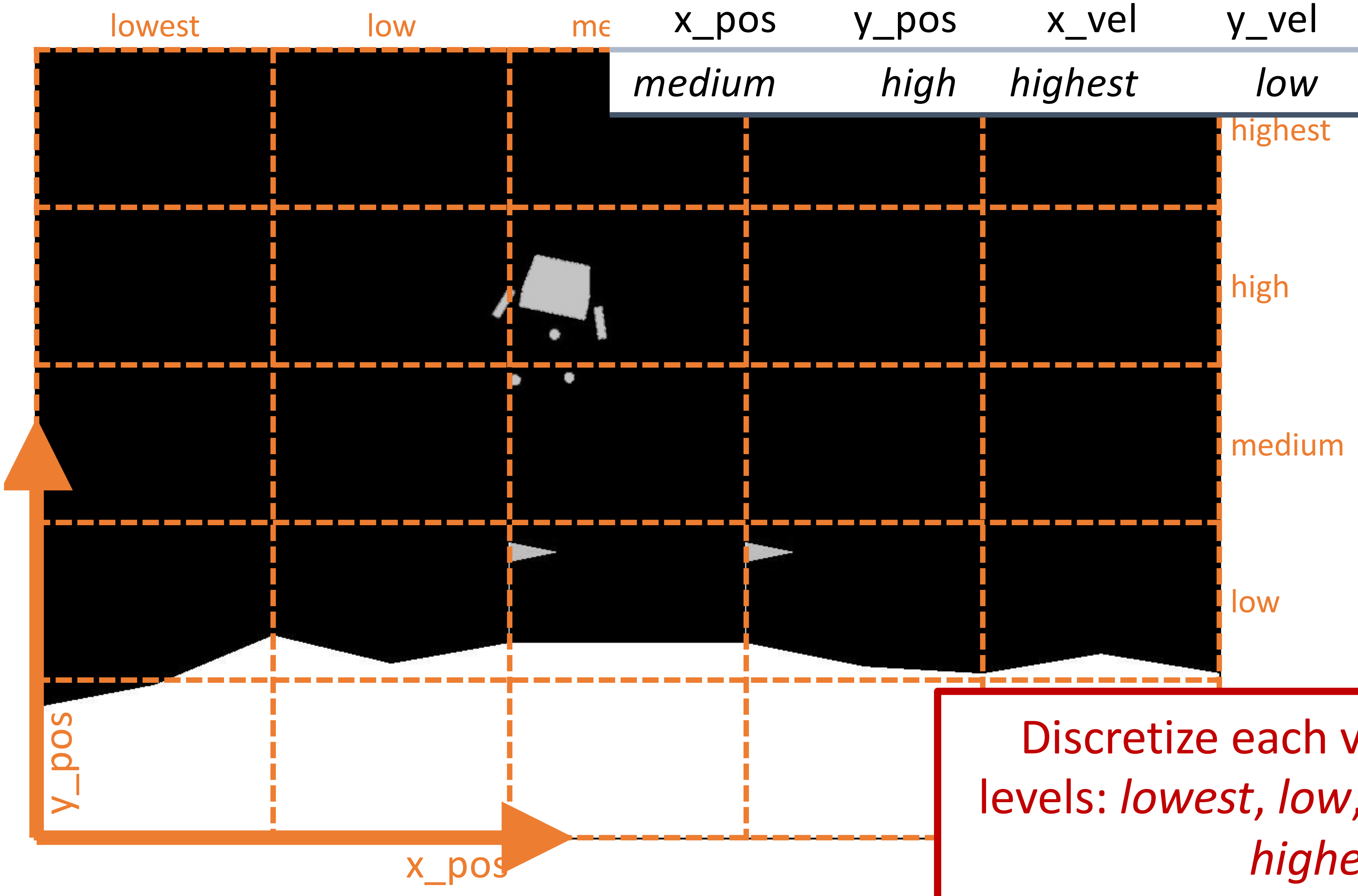


State variables

- position of the spaceship (in x and y coordinates)
- the velocity of the spaceship (in x and y directions),
- the angular velocity of the spaceship
- the angle of the line connecting the spaceship to the landing pad

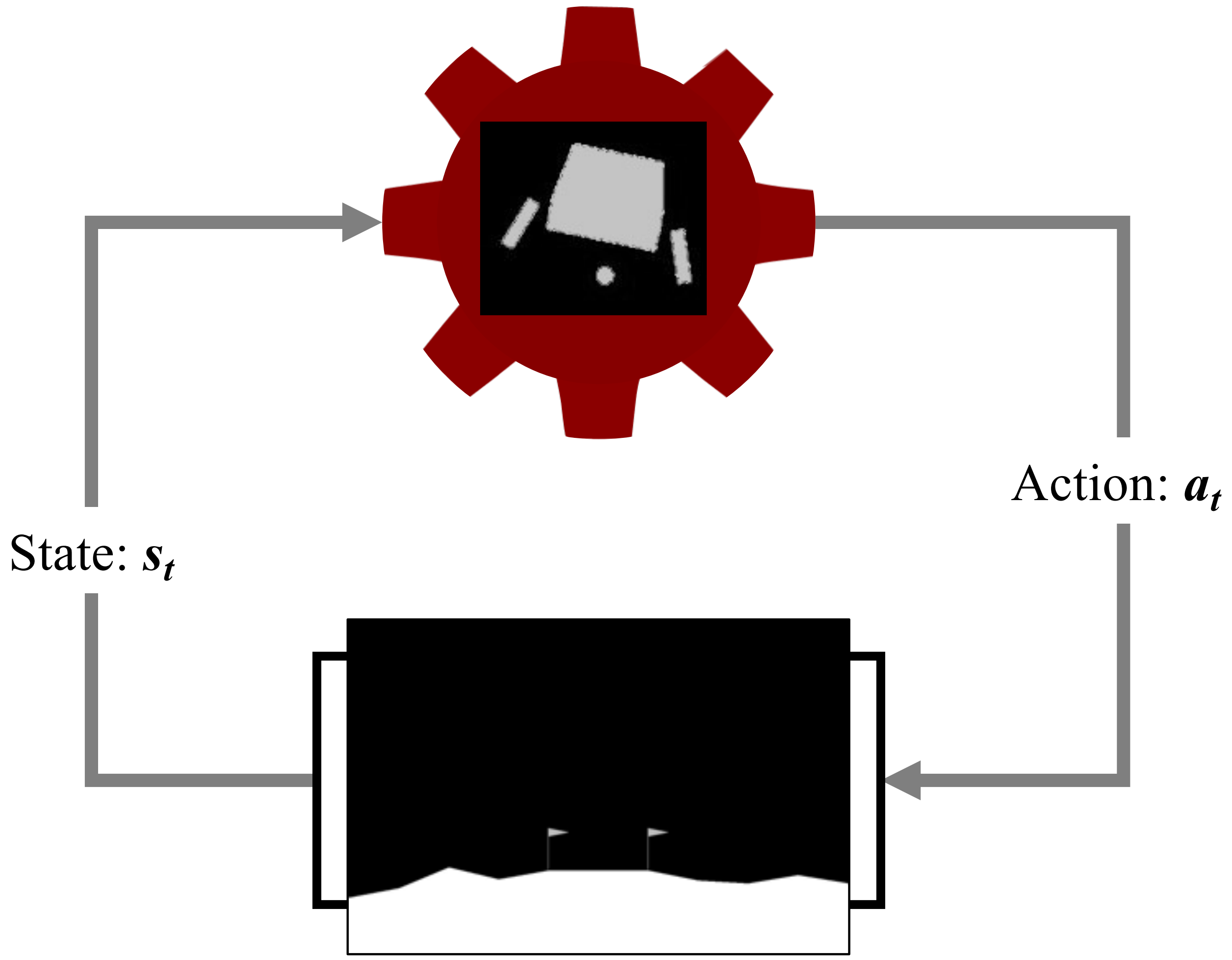


Discretize each value to just 5 levels: *lowest*, *low*, *medium*, *high*, *highest*

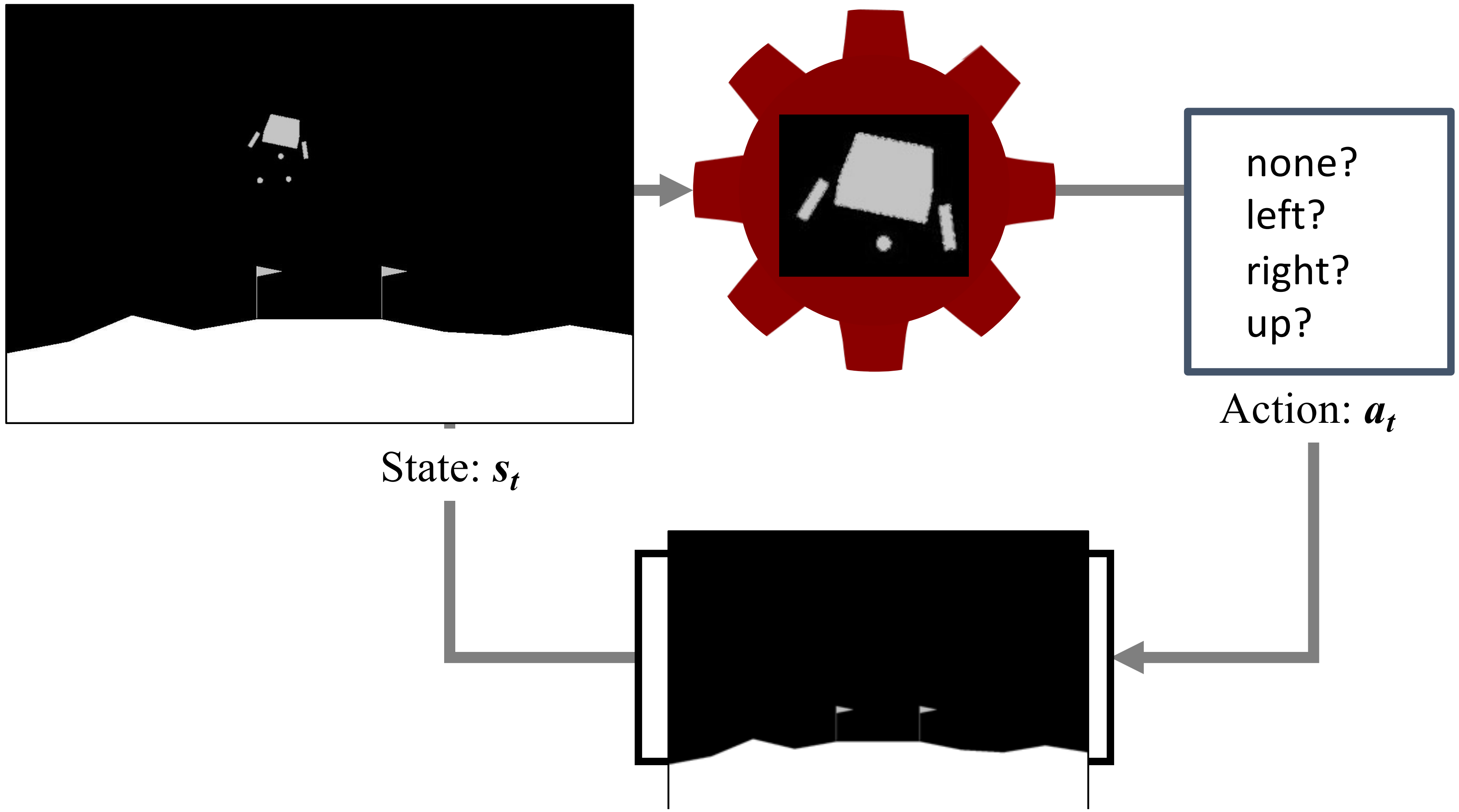


x_pos	y_pos	x_vel	y_vel	ang_vel	angle
medium	*high*	*highest*	*low*	*lowest*	*highest*

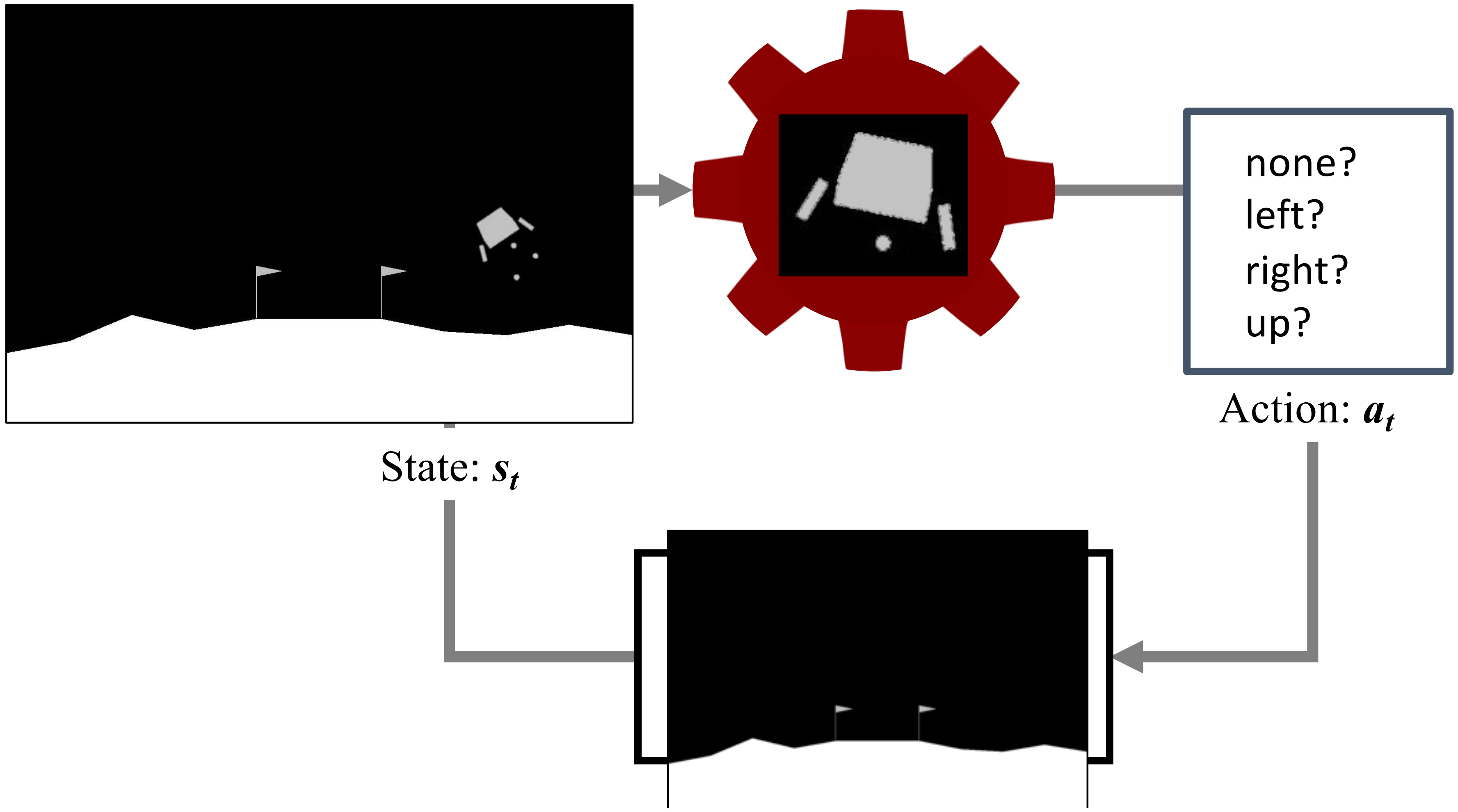
Discretize each value to just 5 levels: *lowest, low, medium, high, highest*


State: s_t
Action: a_t

x_pos	y_pos	x_vel	y_vel	ang_vel	angle
medium	*high*	*highest*	*low*	*lowest*	*highest*



x_pos	y_pos	x_vel	y_vel	ang_vel	angle
high	*low*	*low*	*high*	*medium*	*low*



How Can We Learn This?

Expert Systems

- Interview a set of **experts** who know how to do this
- Encode what we learn from this into a set of rules to control the lander

Supervised Learning

- Get experts to play the game and record a **dataset** of their state-action combinations
- Train a supervised machine learning model that can control the lander using this dataset

alphago

Reinforcement Learning

- Deploy an agent to try this task a lot of times and allow it to train itself through trial and error
- Needs a signal of success and failure - **reward**

alphazero

https://www.youtube.com/watch?v=7-g6OgcbsQ8

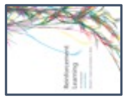
INTRODUCING REINFORCEMENT LEARNING

Reinforcement Learning

"**Reinforcement learning** is learning what to do - how to map situations to actions - so as to maximize a numerical reward signal."

Sutton & Barto

"Reinforcement Learning An Introduction", 2nd Edition, Richard S. Sutton and Andrew G. Barto, MIT Press, 2018. www.incompleteideas.net/book/the-book-2nd.html



Reinforcement Learning

The learner is not told which actions to take, but must discover which actions yield the most reward by trying them

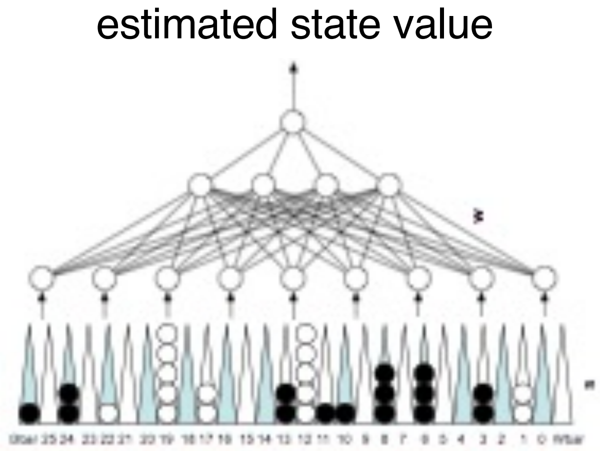
- Often actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards

The two most important distinguishing features of reinforcement learning are:

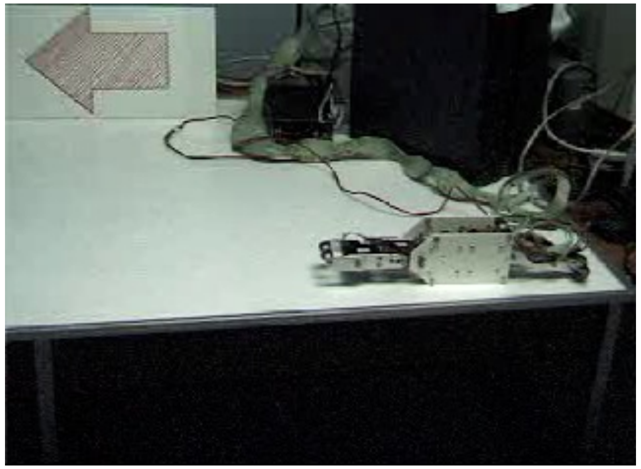
- trial-and-error search
- delayed reward

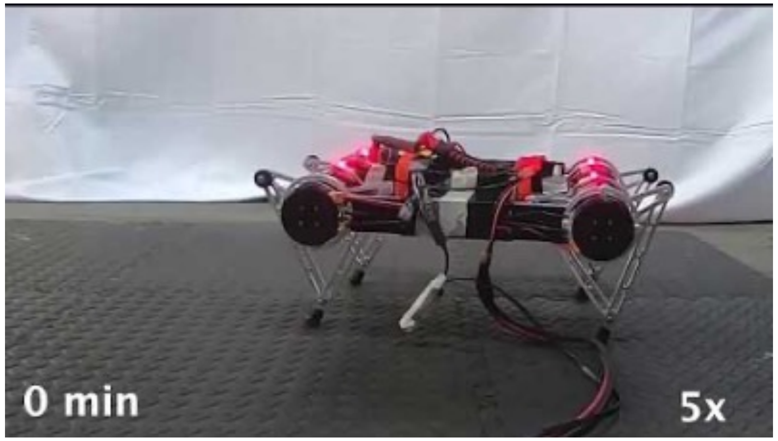
A (Whirlwind) History of Reinforcement Learning

Donal Michie's MENACE (1960s)



TD-Gammon (1990s)


Kimura's Robots (1990s)



RL Robot Gaits (2019)

DeepMind Atari (2015)

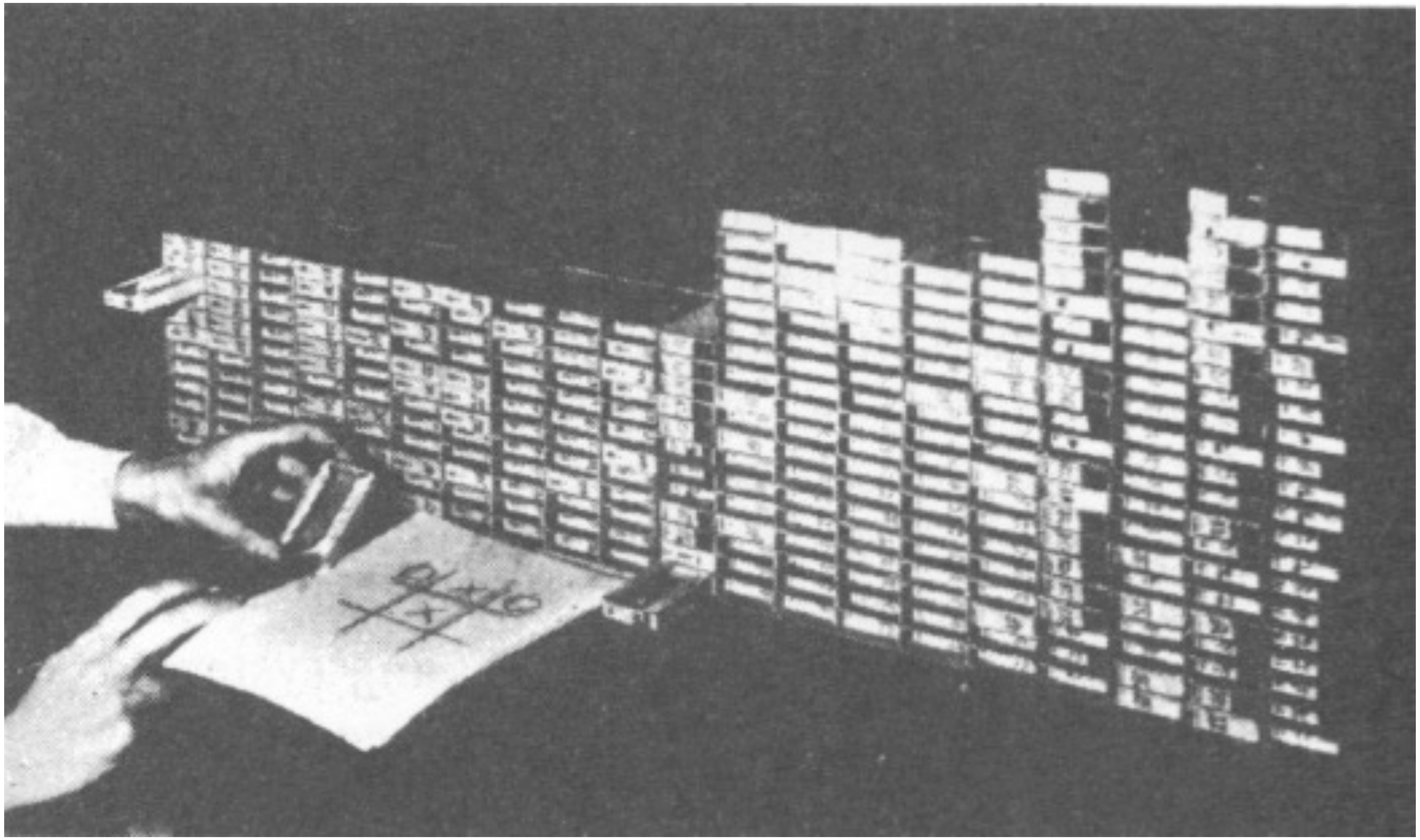
DeepMind AlphaGo (2015)

Donal Michie's MENACE (1960s)

- In the 1960s University of Edinburgh academic Donald Michie built MENACE, a *machine* that could *learn* to play the game of Xs & Os (or noughts and crosses, or tic-tac-toe)
- Constructed from 304 matchboxes carefully filled with precise numbers of collared beads
- With the help of a human operator mindlessly following some simple rules MENACE could play Xs & Os and learn to get better at it!

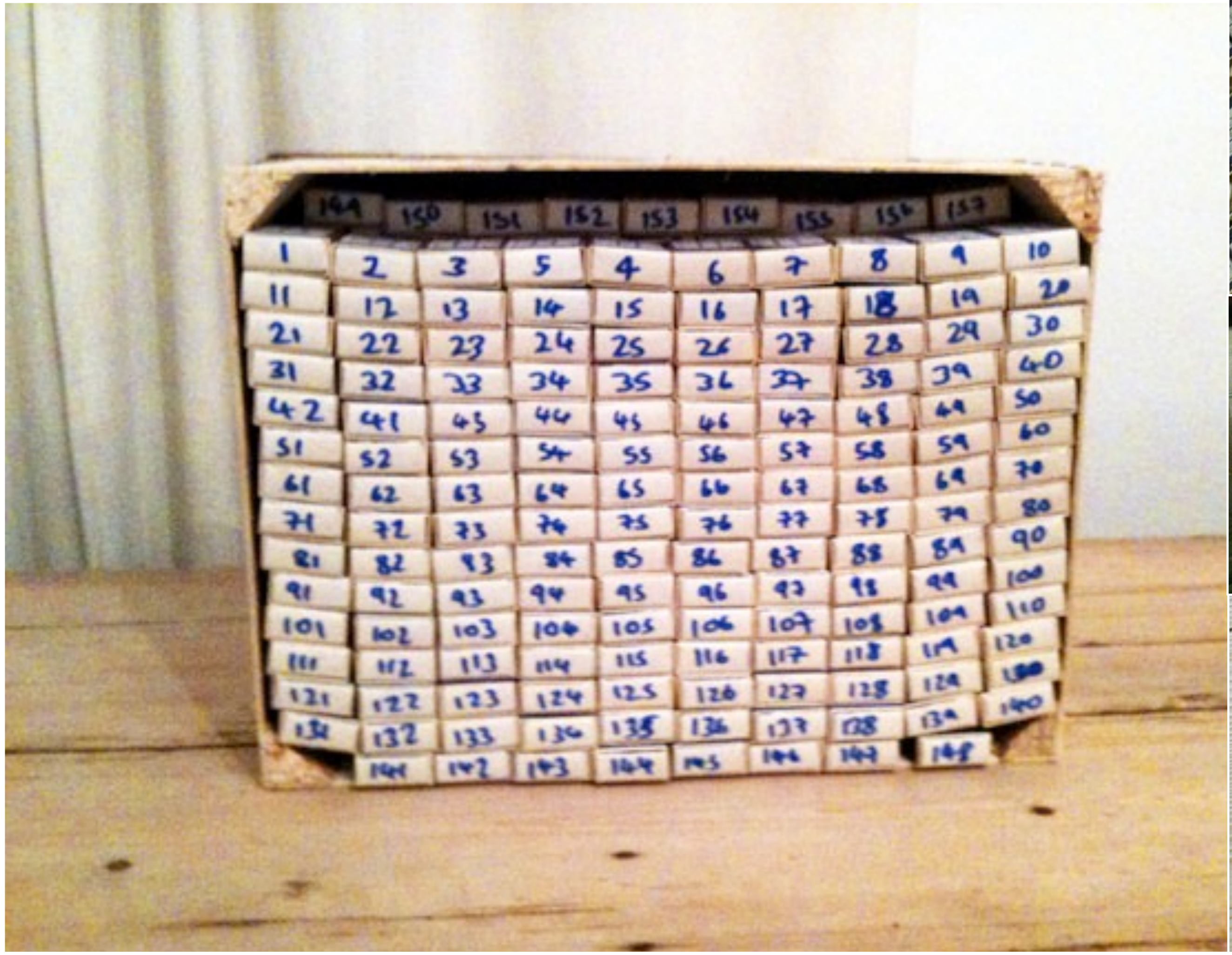
Future of Robotics and Machine Learning: Machine Learning Explained, Rodney Brooks, 2017
https://rodneybrooks.com/forai-machine-learning-explained/

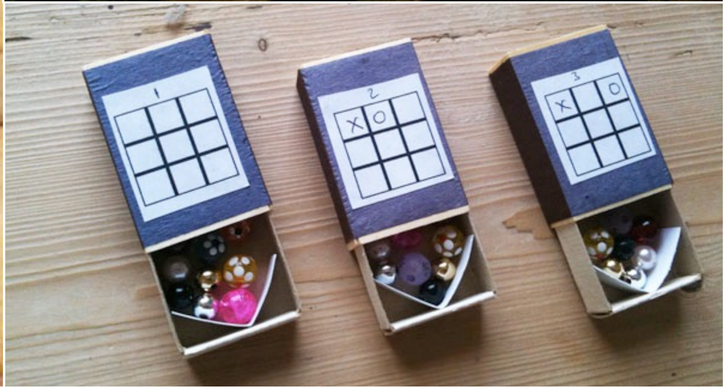
Donal Michie’s MENACE (1960s)



Future of Robotics and Machine Learning: Machine Learning Explained, Rodney Brooks, 2017
https://rodneybrooks.com/forai-machine-learning-explained/

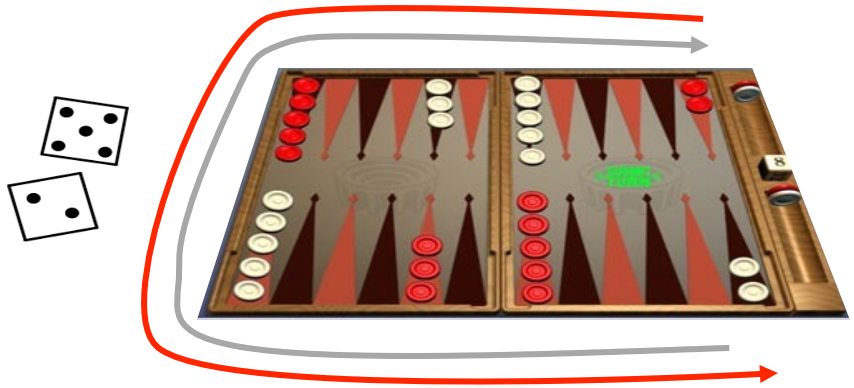
Donal Michie's MENACE (1960s)

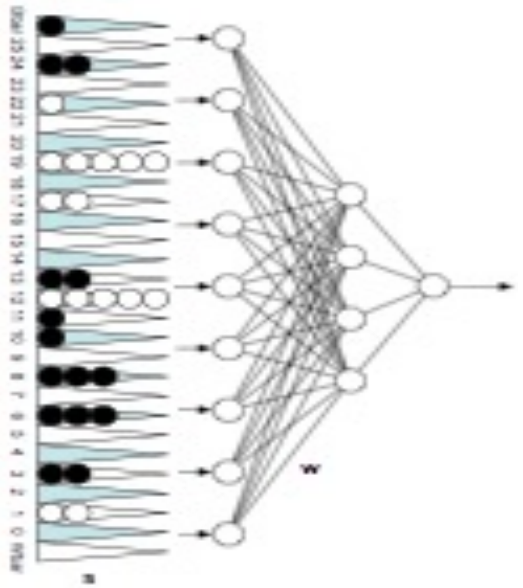




A New THEORY of AWESOMENESS and MIRACLES, Being NOTES and SLIDES on a talk given at PLAYFUL 09, concerning CHARLES BABBAGE, HEATH ROBINSON, MENACE and MAGE, 2009, James BRIDLE http://shorttermmemoryloss.com/menace/

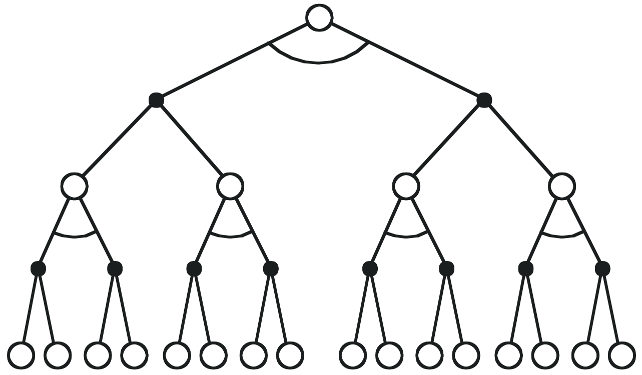
TD-Gammon (1990s)





estimated state value
(≈ prob of winning)

Action selection
by a shallow search



Start with a random network

Play millions of games against itself

Learn a value function from this simulated experience

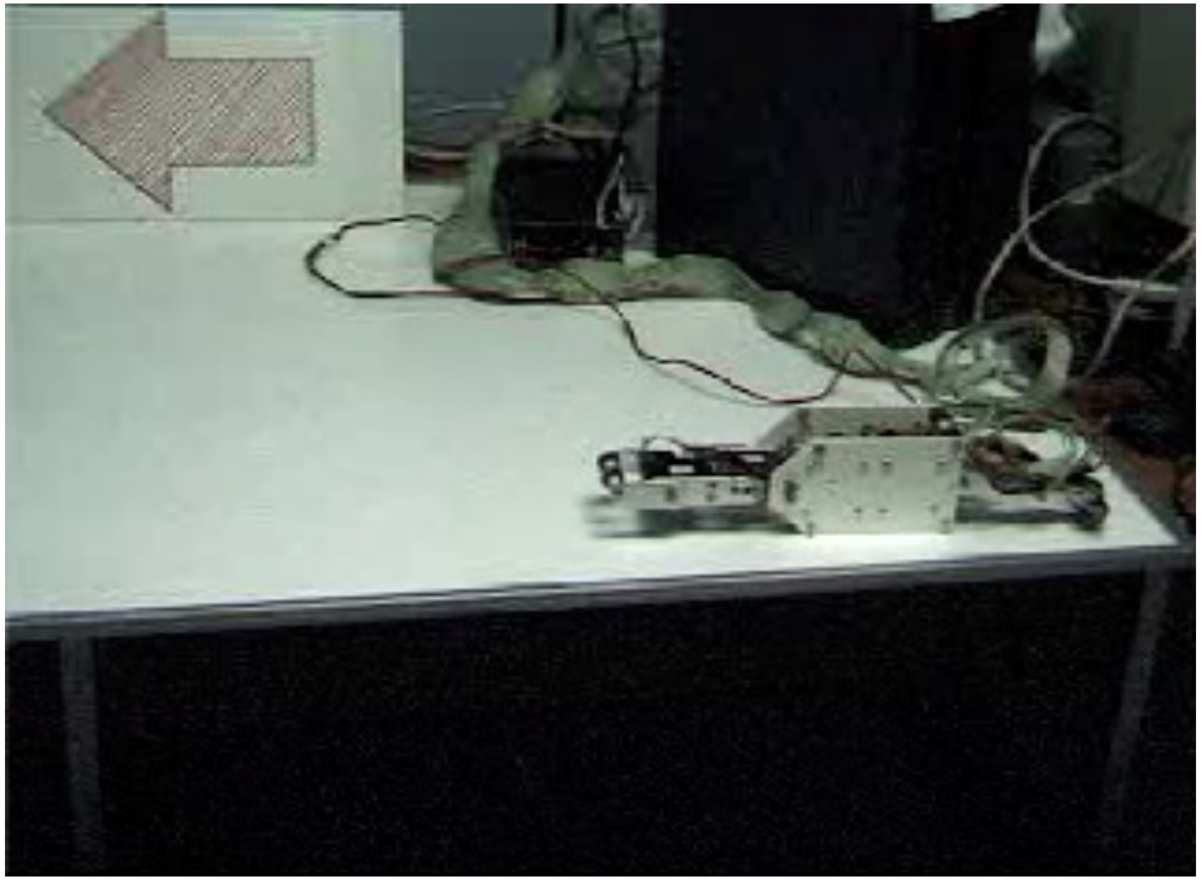
6 weeks later it is the best player of backgammon in the world!

Originally used expert, handcrafted features, later repeated with raw board positions

Tesauro, Gerald. "Temporal difference learning and TD-Gammon." *Communications of the ACM* 38.3 (1995): 58-68. http://enzodesiage.com/wp-content/uploads/2017/08/tesauro-tdgammon-1995.pdf

Hajime Kimura’s RL Robots (1990s)

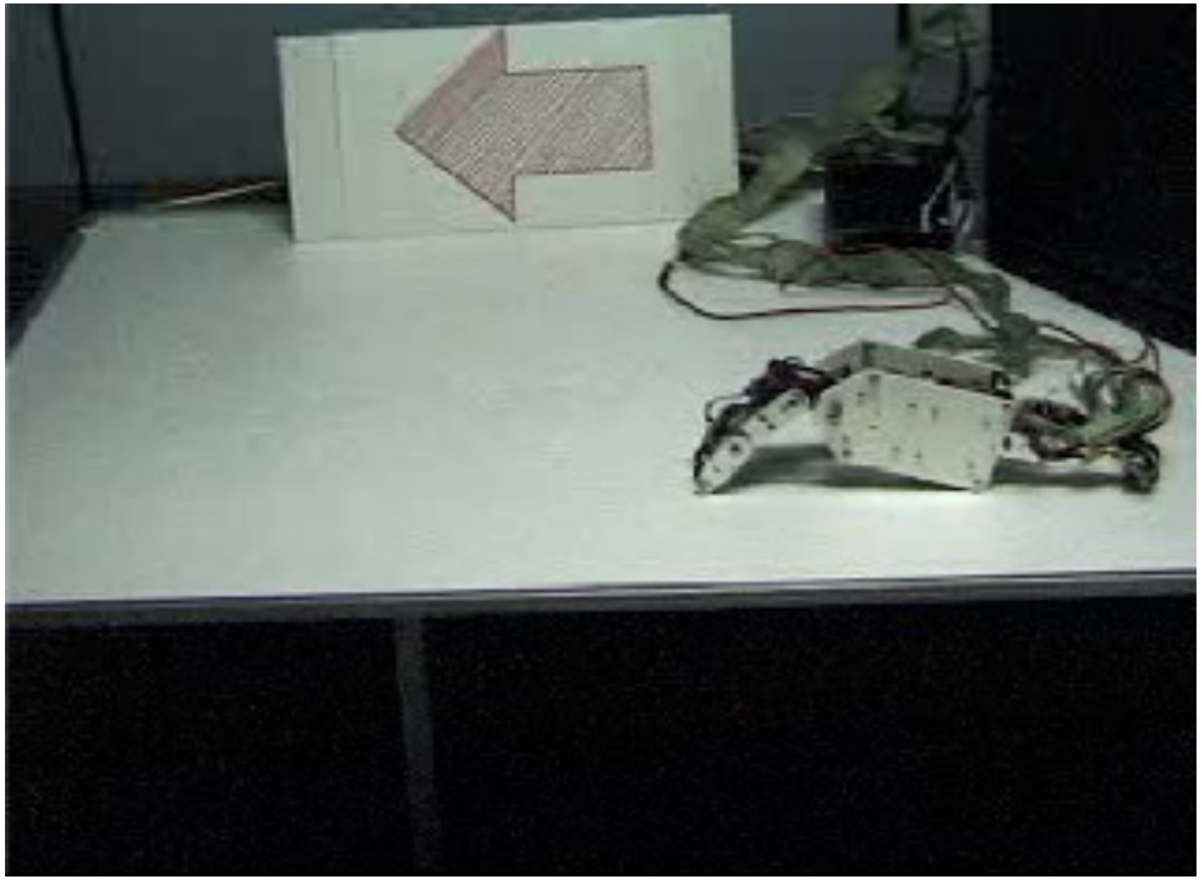
- Small robots composed of a simple two link arm controlled with two servo motors
- A small wheel sensor to record movement is included
- The goal is to learn control rules to move to the front or back - a move is taken about every 0.2 sec
- The body's movement for each time step is given to the agent as an instantaneous reward



Reinforcement Learning using Stochastic Gradient Algorithm and its Application to Robots, The Transaction of the Institute of Electrical Engineers of Japan, Vol.119, No.8 (1999) Hajime Kimura, Shigenobu Kobayashi http://sysplan.nams.kyushu-u.ac.jp/gen/papers/denki99.pdf (in Japanese!)

Hajime Kimura’s RL Robots (1990s)

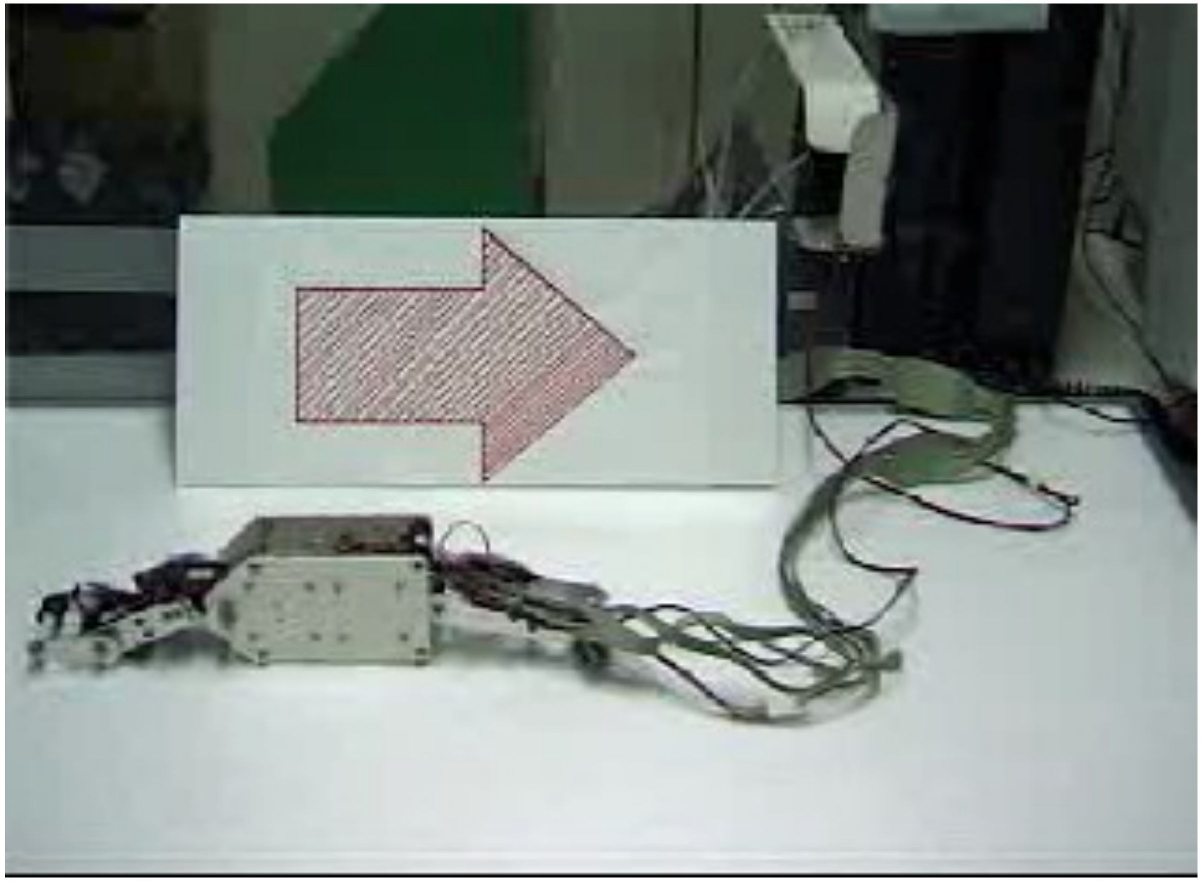
- Small robots composed of a simple two link arm controlled with two servo motors
- A small wheel sensor to record movement is included
- The goal is to learn control rules to move to the front or back - a move is taken about every 0.2 sec
- The body's movement for each time step is given to the agent as an instantaneous reward



Reinforcement Learning using Stochastic Gradient Algorithm and its Application to Robots, The Transaction of the Institute of Electrical Engineers of Japan, Vol.119, No.8 (1999) Hajime Kimura, Shigenobu Kobayashi http://sysplan.nams.kyushu-u.ac.jp/gen/papers/denki99.pdf (in Japanese!)

Hajime Kimura’s RL Robots (1990s)

- Small robots composed of a simple two link arm controlled with two servo motors
- A small wheel sensor to record movement is included
- The goal is to learn control rules to move to the front or back - a move is taken about every 0.2 sec
- The body's movement for each time step is given to the agent as an instantaneous reward



Reinforcement Learning using Stochastic Gradient Algorithm and its Application to Robots, The Transaction of the Institute of Electrical Engineers of Japan, Vol.119, No.8 (1999) Hajime Kimura, Shigenobu Kobayashi http://sysplan.nams.kyushu-u.ac.jp/gen/papers/denki99.pdf (in Japanese!)

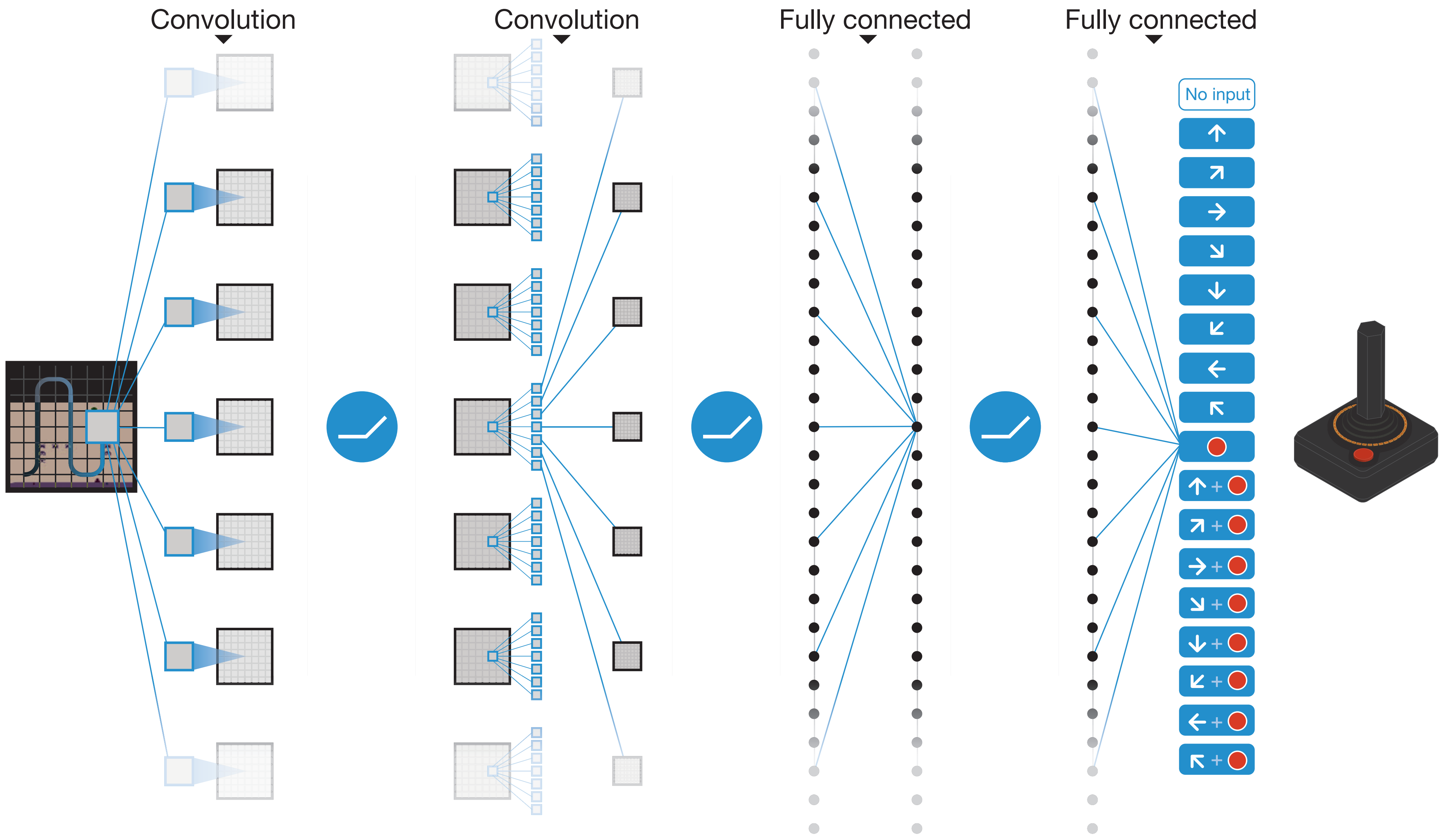
Deep Reinforcement Learning Applied to Classic Atari Video Games (2010s)

Space Invaders

Breakout

Enduro

https://www.youtube.com/watch?v=V1eYniJ0Rnk


Convolution
Convolution
Fully connected
Fully connected
No input

Deep Reinforcement Learning Applied to Classic Atari Video Games (2010s)

- Learned to play 49 games for the Atari 2600 game console, without labels or human input, from self-play and the score alone
- Learned to play better than all previous algorithms and at human level for more than half the games
- Same learning algorithm applied to all 49 games w/o human tuning!

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G. and Petersen, S., 2015. Human-level control through deep reinforcement learning. *Nature*, *518*(7540), p.529. http://www.davidqiu.com:8888/research/nature14236.pdf

DeepMind AlphaGo (2015)

Go was long considered one of the most difficult games for which to build computer players:

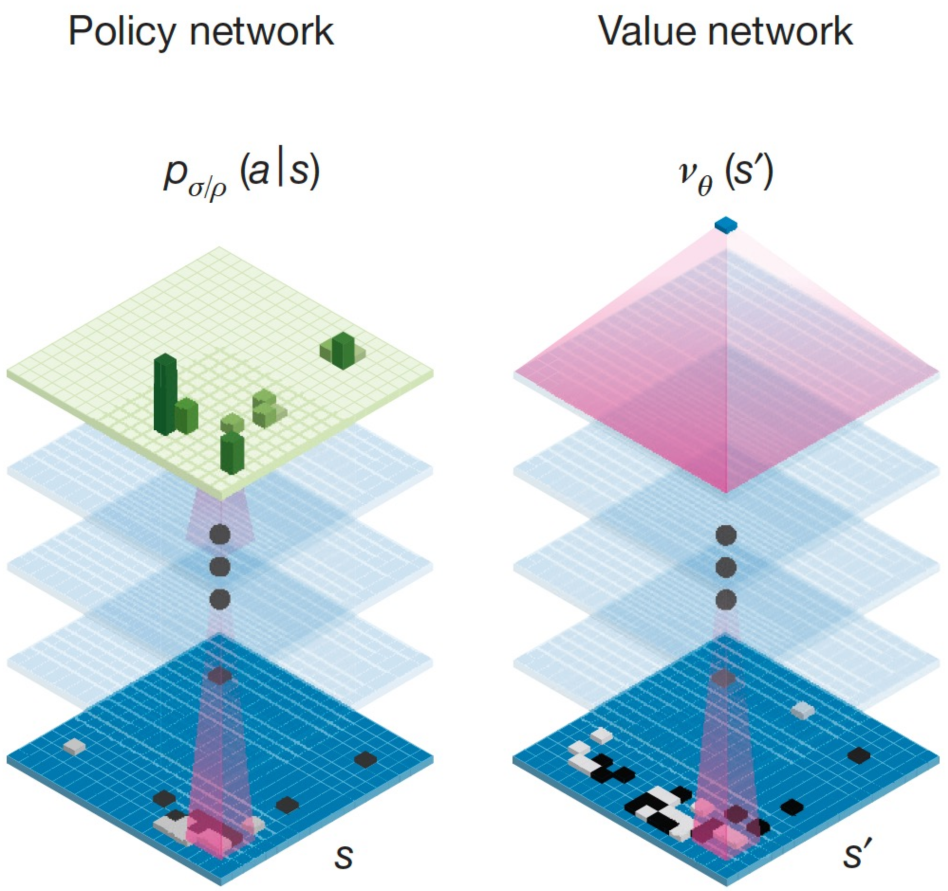
- Massive branching factor
- Difficult to write heuristics to evaluate board positions

In 2015 the DeepMind AlphaGo system beat Lee Sedol – the Gary Kasporov of Go!

Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M. and Dieleman, S., 2016. Mastering the game of Go with deep neural networks and tree search. *nature*, *529*(7587), pp.484-489. https://5y1.org/download/262ac79a4eb3d55c6cedae1005a156a5.pdf

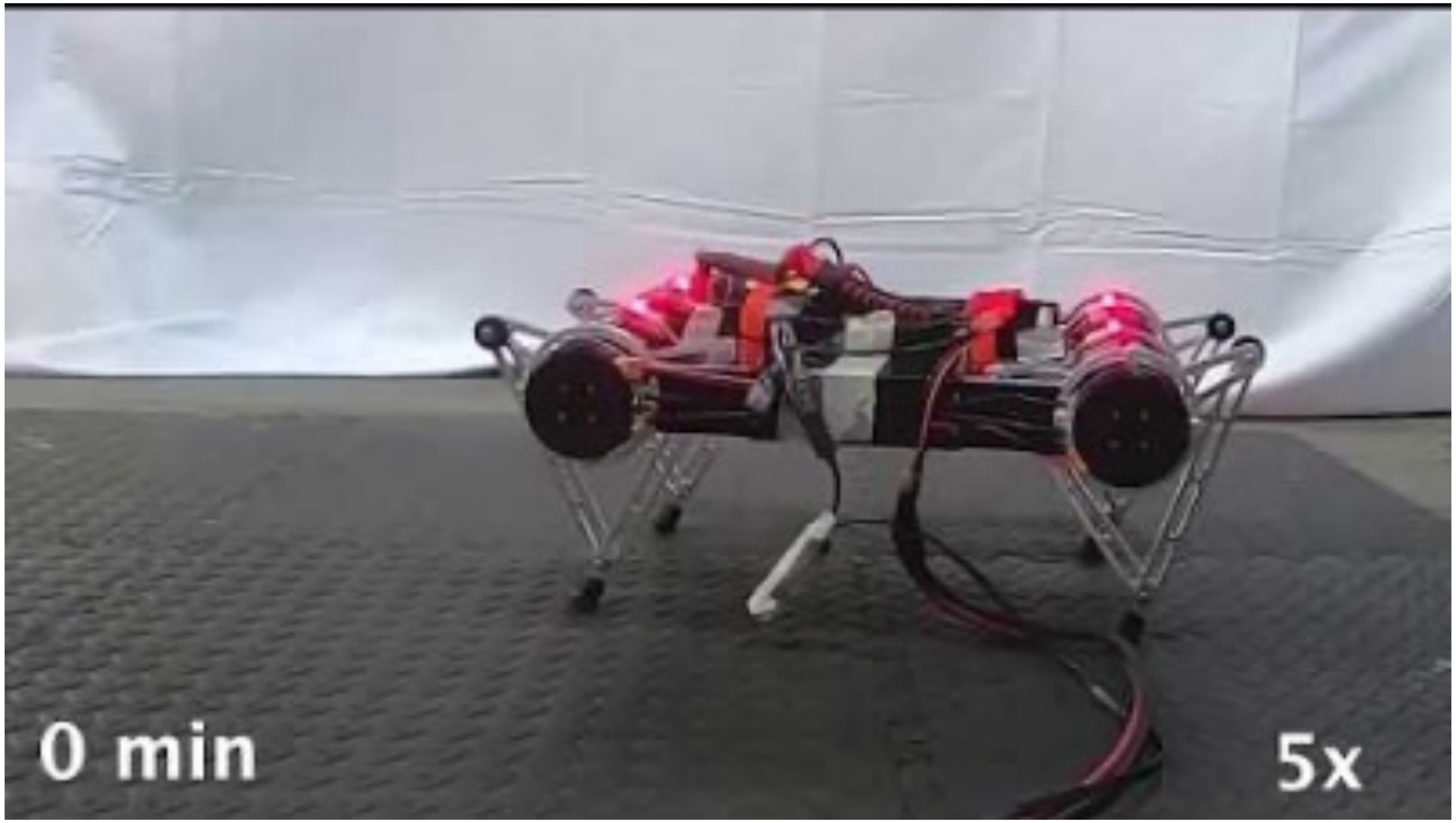
DeepMind AlphaGo (2015)

AlphaGo used a combination of tree search supervised learning and reinforcement learning to build an expert go player



Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M. and Dieleman, S., 2016. Mastering the game of Go with deep neural networks and tree search. *nature*, *529*(7587), pp.484-489. https://5y1.org/download/262ac79a4eb3d55c6cedae1005a156a5.pdf

RL For Robot Gaits



https://www.youtube.com/watch?v=n2gE7n11h1Y

Learning to Walk via Deep Reinforcement Learning. Tuomas Haarnoja, Sehoon Ha, Aurick Zhou, Jie Tan, George Tucker, Sergey Levine. Robotics: Science and Systems (RSS). 2019. https://arxiv.org/abs/1812.11103

Reinforcement Learning vs Supervised Learning

- **Supervised learning** requires a training set of labelled instances provided by a knowledgeable external supervisor
  - Each instance is a description of a situation together with the correct action the system should take
  - The object is for the system to generalize its responses so that it acts correctly in situations not present in the training set
- In interactive problems it is often impractical to obtain these labelled training sets
  - In uncharted territory an agent must be able to learn from its own experience

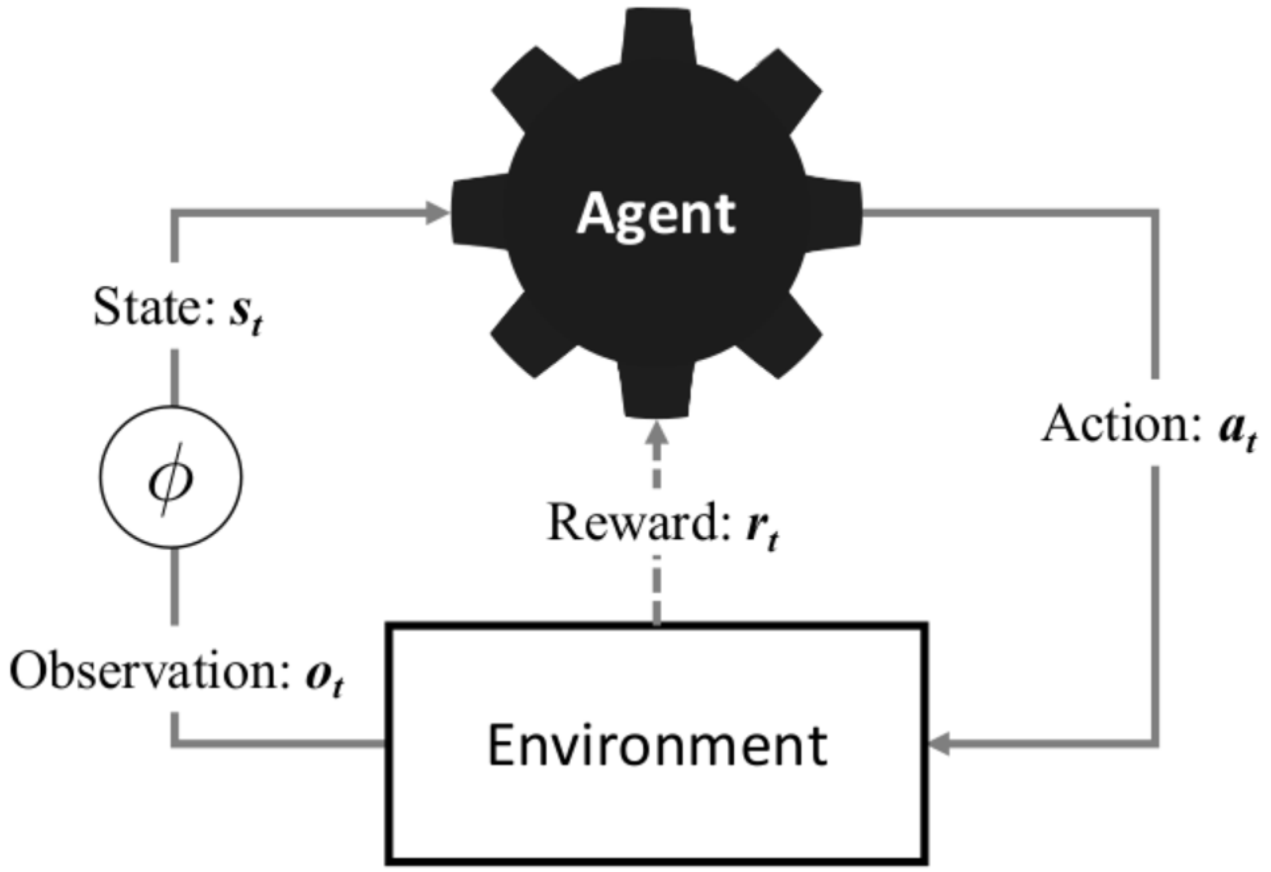
Reinforcement Learning vs Unsupervised Learning

Unsupervised learning is typically about finding structure hidden in collections of unlabelled data

Although reinforcement learning does not rely on examples of correct behaviour it is not unsupervised

Reinforcement learning maximizes a **reward signal** which is a type of supervision

Reinforcement Learning



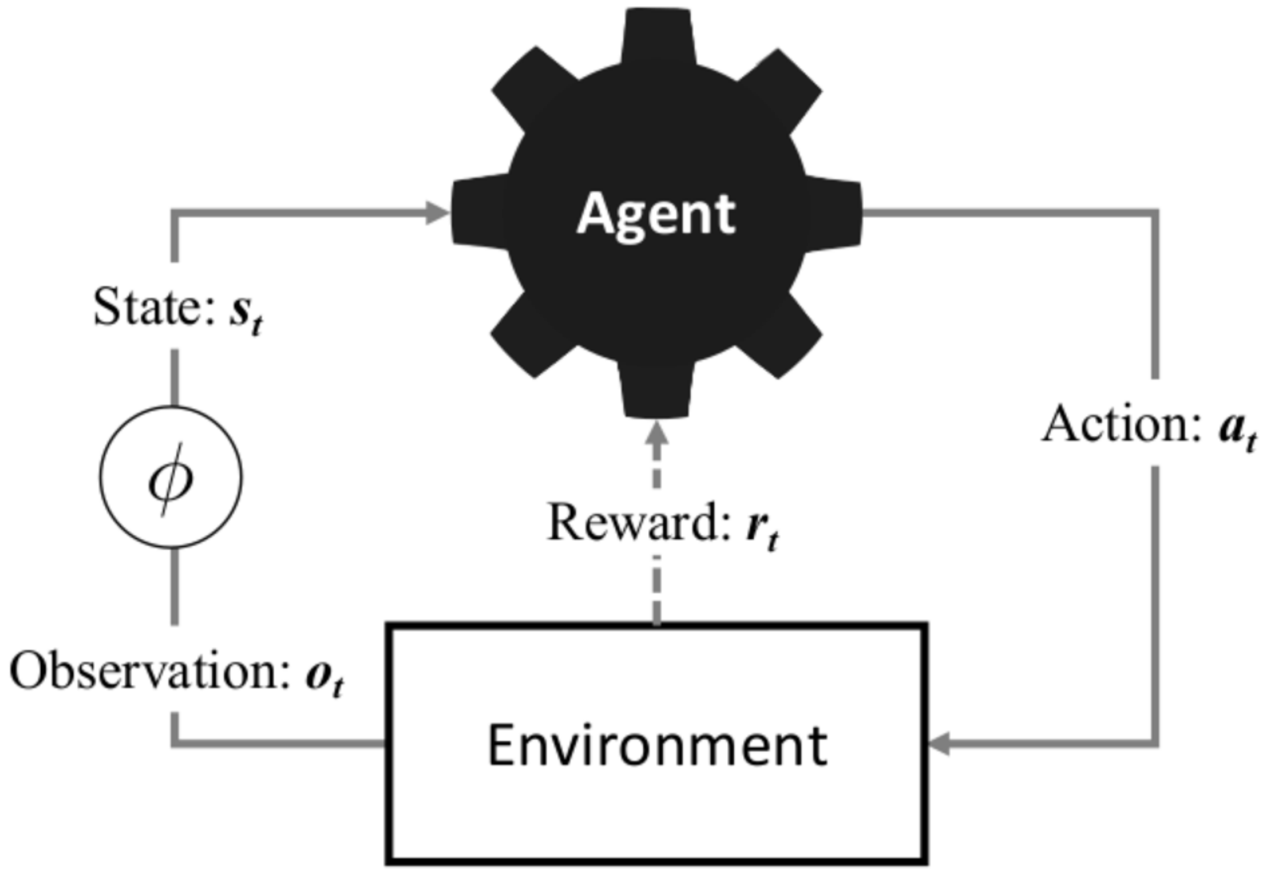
SUMMARY

Summary

We have introduced the key ideas in reinforcement learning

- an agent
- an environment
- a policy A strategy or rule that the agent follows to decide actions.
- a reward signal A numerical feedback from the environment.
- a value function Measures how good a state is in the long term.
- a model of the environment (optional)

Reinforcement Learning



Summary

Over the coming lectures we will expand our discussion to cover key ideas in reinforcement learning:

- *k*-armed bandit problems
- Markov decision processes
- Temporal difference learning
- SARSA
- Q learning
- Deep Q learning
- Actor Critic Methods

Questions

?