MIT DEEP REINFORCEMENT LEARNING LECTURE: SUMMARY AND TAKEAWAYS

Scott O'Hara

Metrowest Developers Machine Learning Group

03/06/2019

DEEP REINFORCEMENT LEARNING LECTURE AT MIT(JAN 22, 2019)

- Prof. Lex Fridman
- Lecture is part of an introductory class on deep learning (**6.S091**)
- The course is an Independent Activities Period (IAP) at MIT.
- Website: ***https://deeplearning.mit.edu/***
- Also see: ***http://introtodeeplearning.com/***

LECTURE SUMMARY

1. The 1st quarter of the lecture was spent motivating deep learning and reinforcement learning and giving context in the wider developments and goals in AI

2. Dr Fridman spent the 2nd quarter of the lecture covering general reinforcement learning (RL) that we covered last year, including the grid world example.

3. The second half of the lecture introduced deep reinforcement learning (DRL) where DRL = RL + NN. He surveyed various DRL algorithms and techniques with extended discussion of deep Q-learning (DQN) and AlphaGo.

PERSONAL TAKEAWAYS

1. All machine learning is "supervised" by a ***loss function*** where supervised learning is described as "learning by example" and reinforcement learning is described as "learning by experience".

2. The relationship between deep learning and deep reinforcement learning can be characterized in a kind of robotic process loop framework i.e.,

perceive → represent → learn → action

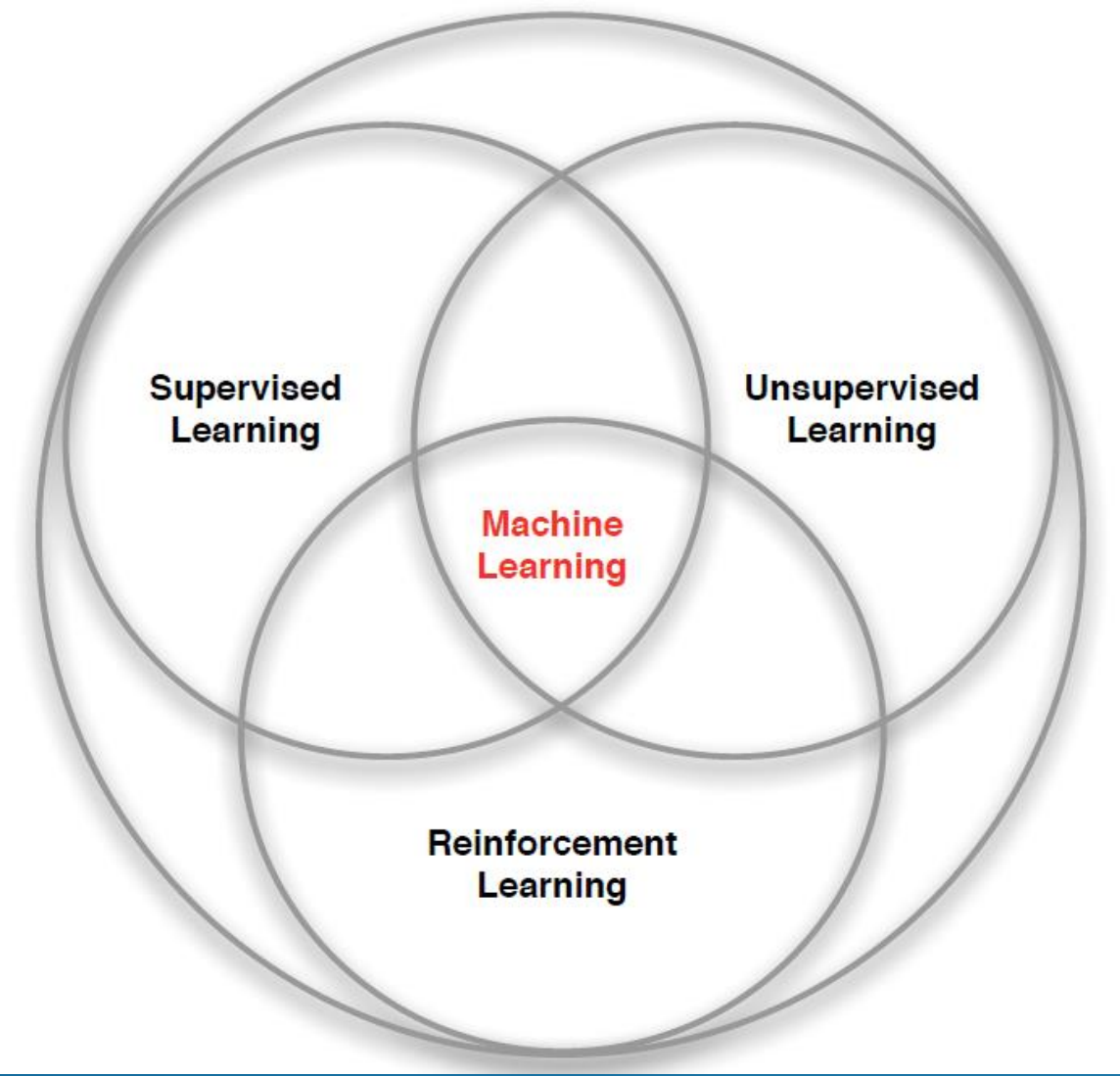
PERSONAL TAKEAWAYS (2)

3. Deep Q-learning can be viewed as approximate Q-learning where the Q(s,a) array is replaced by a deep neural network Q(s,a |dNN).

4. Realistic simulation of the real world is very important for training deep reinforcement learning agents.

5. For the most part, deep learning is NOT being used in current real world robots or driverless cars but this may be changing.

DIFFERENT KINDS OF MACHINE LEARNING



BRANCHES OF MACHINE LEARNING

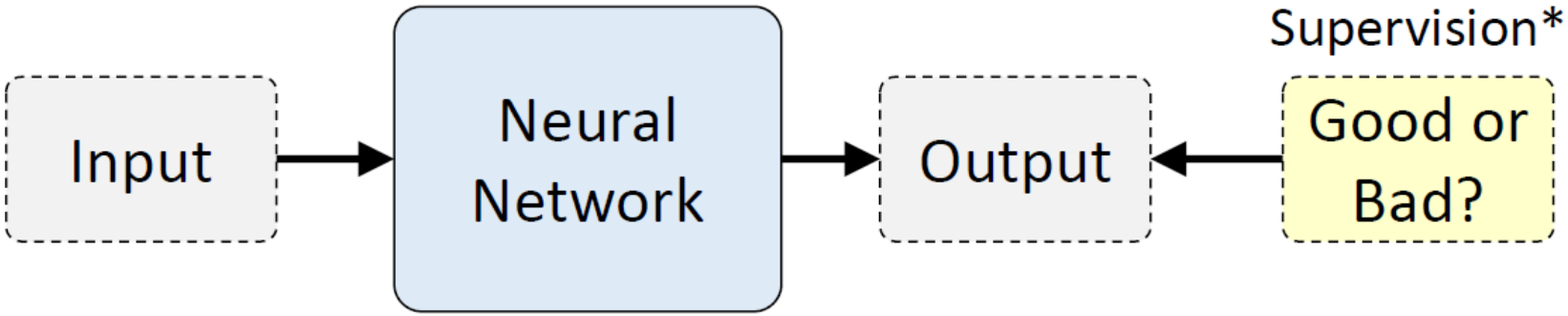
Credit: adapted from lecture slides David Silver, DeepMind, "Introduction to Reinforcement Learning

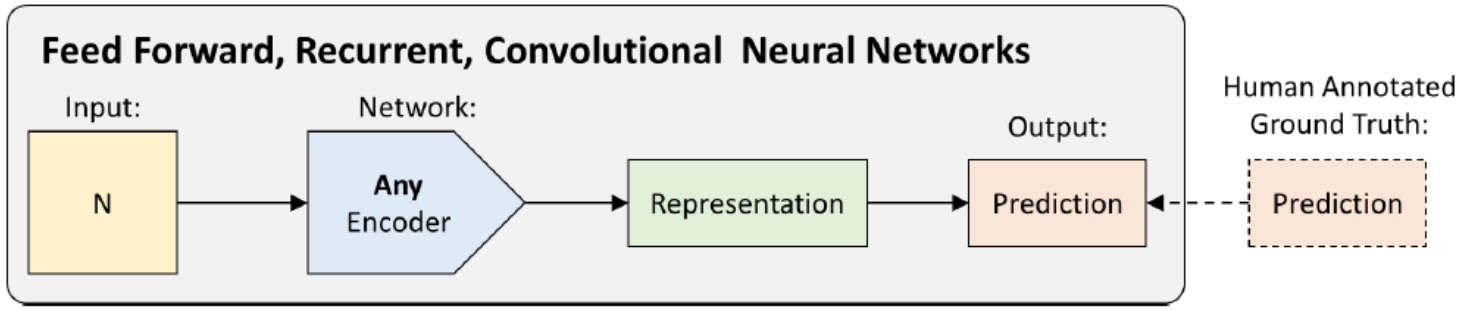
Types of Learning

- Supervised Learning
- Semi-Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

It's all "supervised" by a loss function!

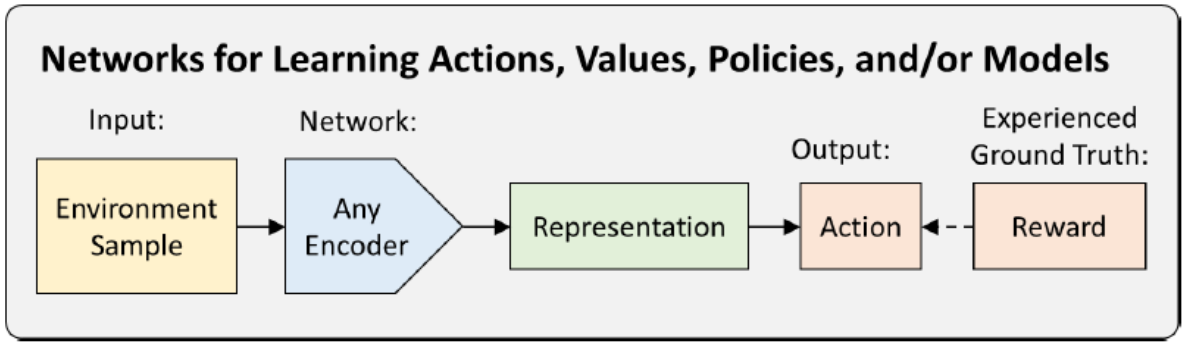
**Someone has to say what's good and what's bad (see Socrates, Epictetus, Kant, Nietzsche, etc.)*





Supervised learning is "teach by **example**":
Here's some examples, now learn patterns in these example.

Reinforcement learning is "teach by **experience**":
Here's a world, now learn patterns by exploring it.



DEEP LEARNING AND DEEP REINFORCEMENT LEARNING

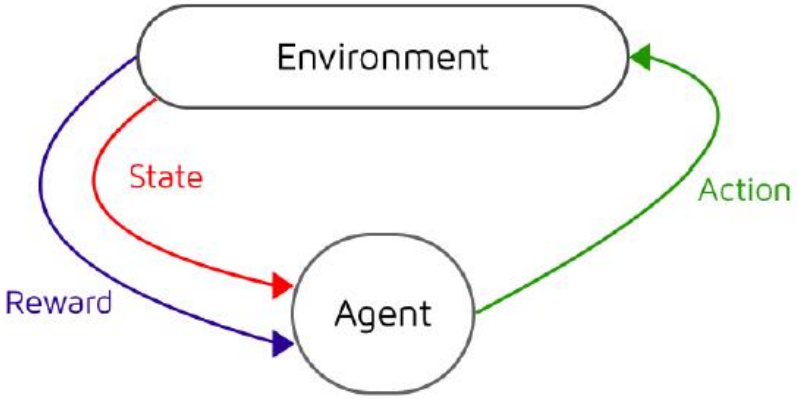
Reinforcement Learning Framework

At each step, the agent:

- Executes action
- Observe new state
- Receive reward

Open Questions:

- What cannot be modeled in this way?
- What are the challenges of learning in this framework?



Environment

Sensors

Sensor Data

Feature Extraction

Representation

Machine Learning

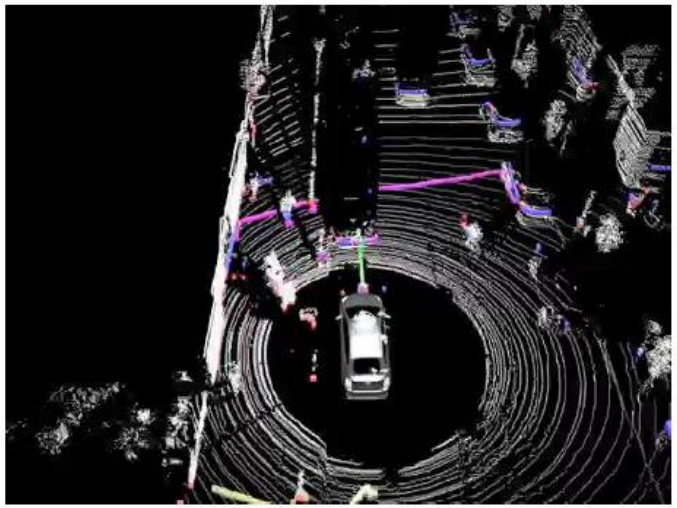
Knowledge

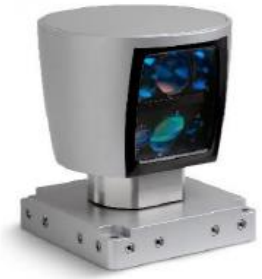
Reasoning

Planning

Action

Effector





Lidar

Camera
(Visible, Infrared)

Radar

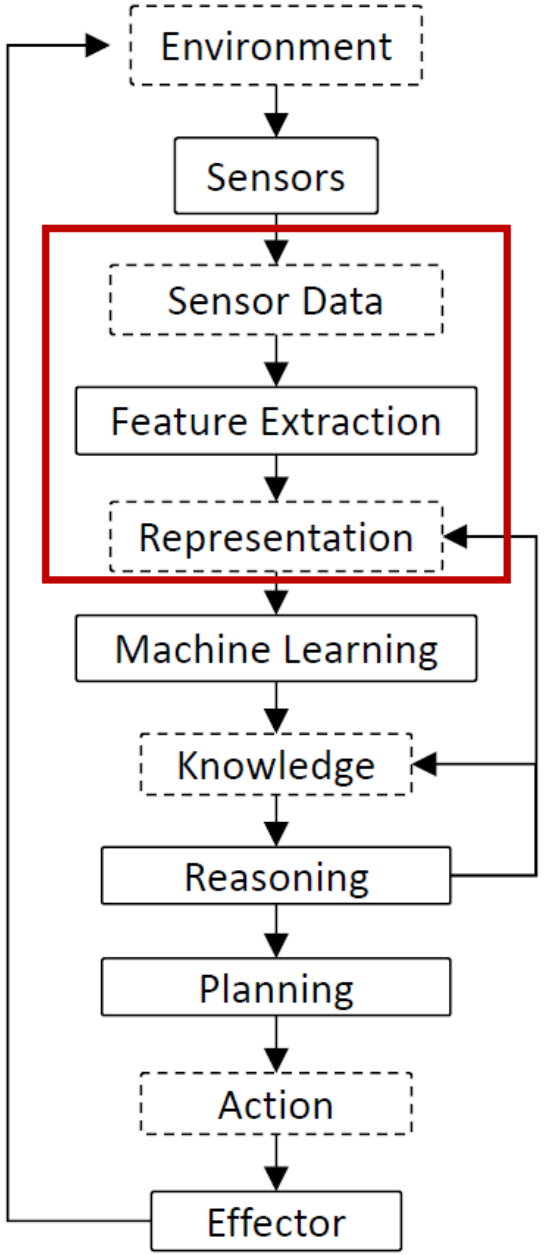
GPS

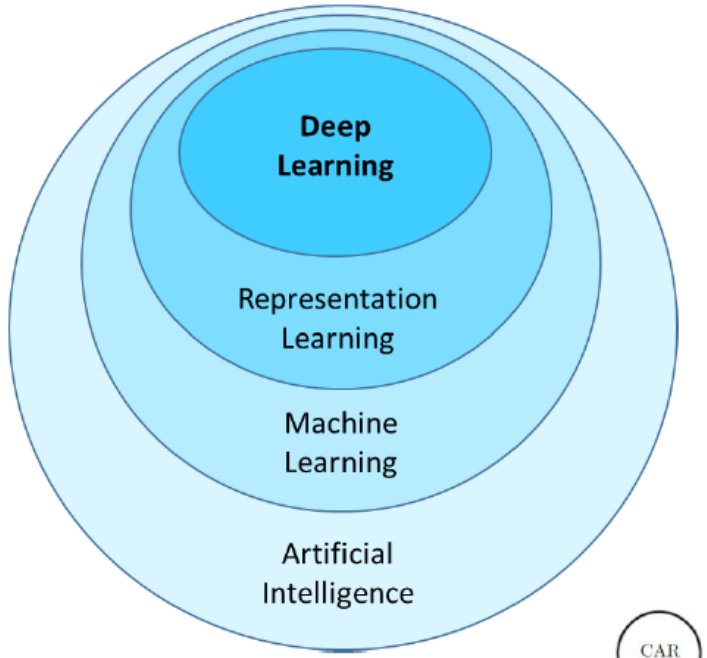
Stereo Camera

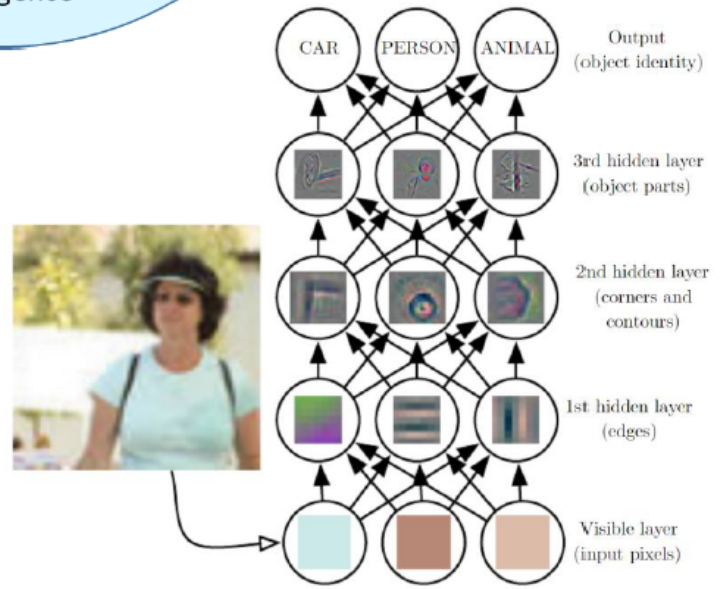
Microphone

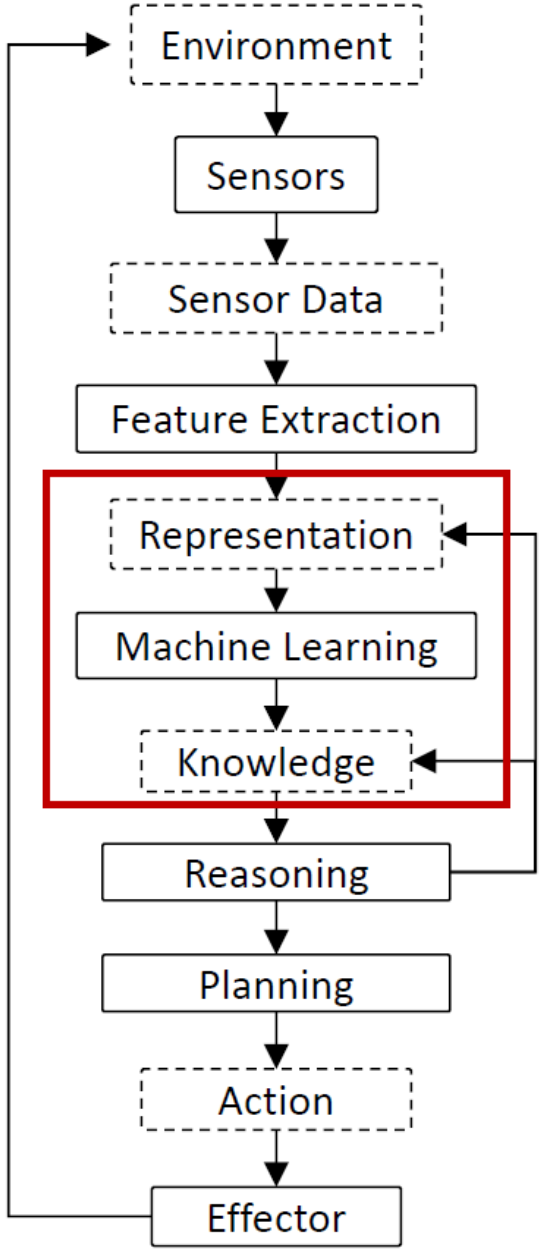
Networking
(Wired, Wireless)

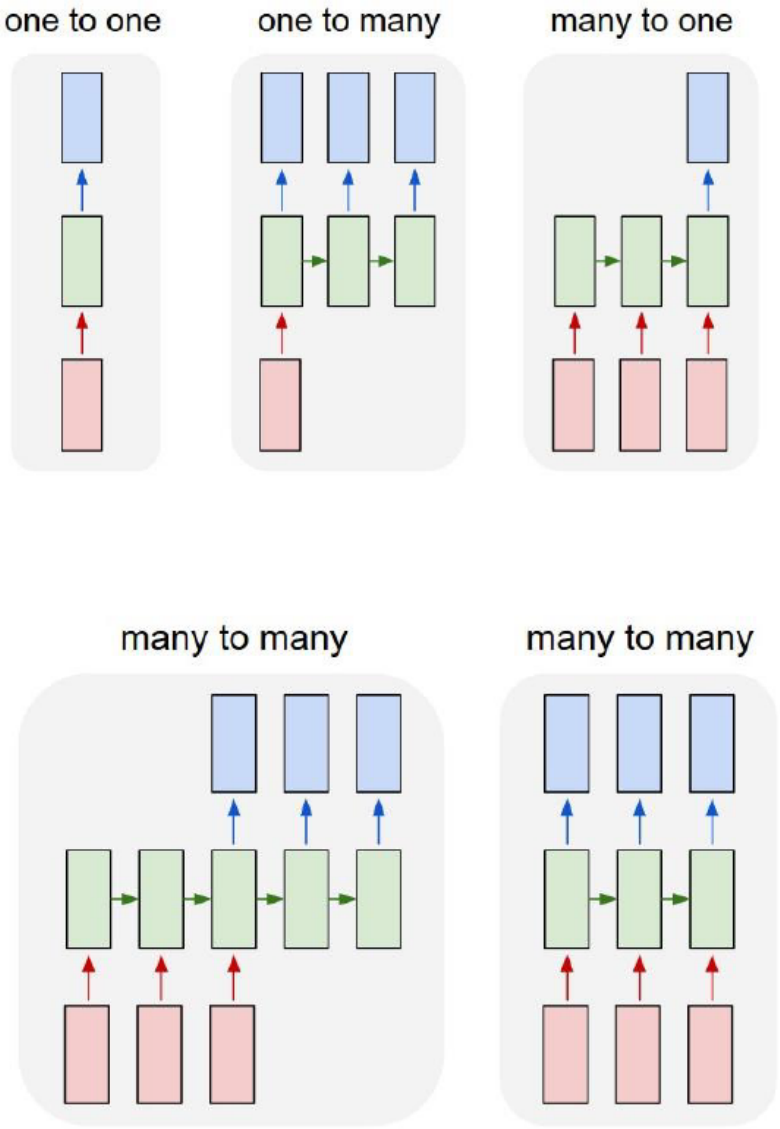
IMU


Environment
Sensors
Sensor Data
Feature Extraction
Representation
Machine Learning
Knowledge
Reasoning
Planning
Action
Effector


Deep Learning
Representation Learning
Machine Learning
Artificial Intelligence


CAR
PERSON
ANIMAL
Output (object identity)
3rd hidden layer (object parts)
2nd hidden layer (corners and contours)
1st hidden layer (edges)
Visible layer (input pixels)


Environment
Sensors
Sensor Data
Feature Extraction
Representation
Machine Learning
Knowledge
Reasoning
Planning
Action
Effector


one to one
one to many
many to one
many to many
many to many

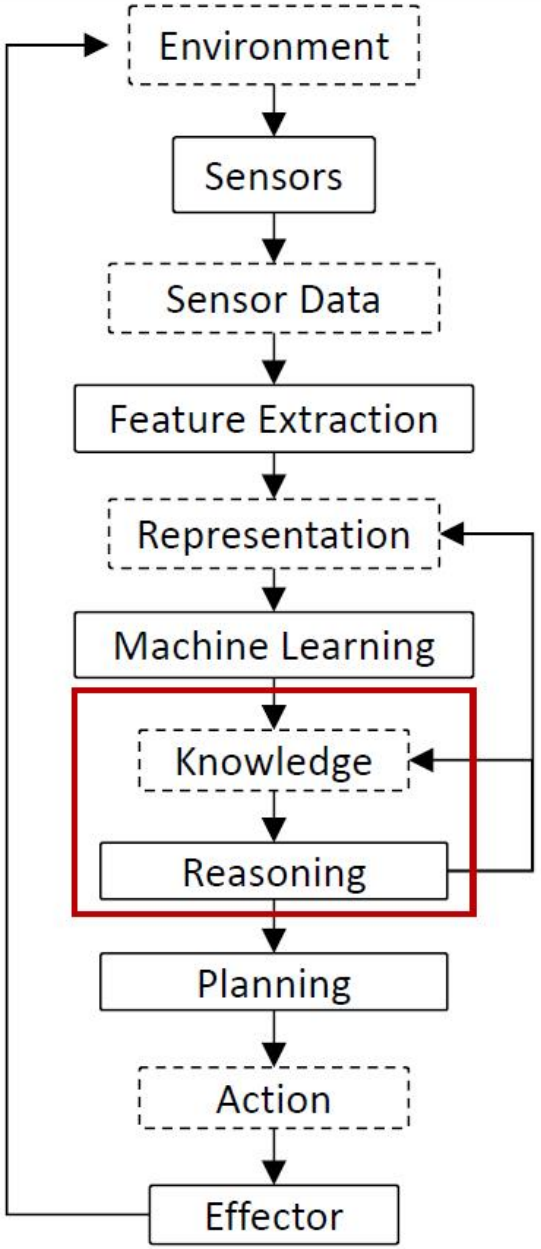
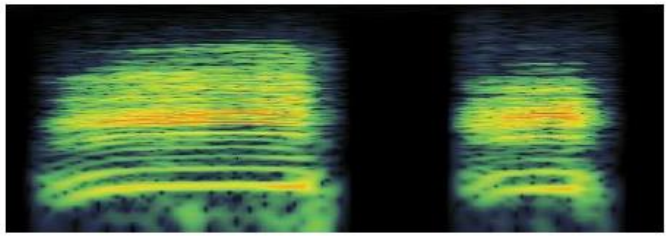
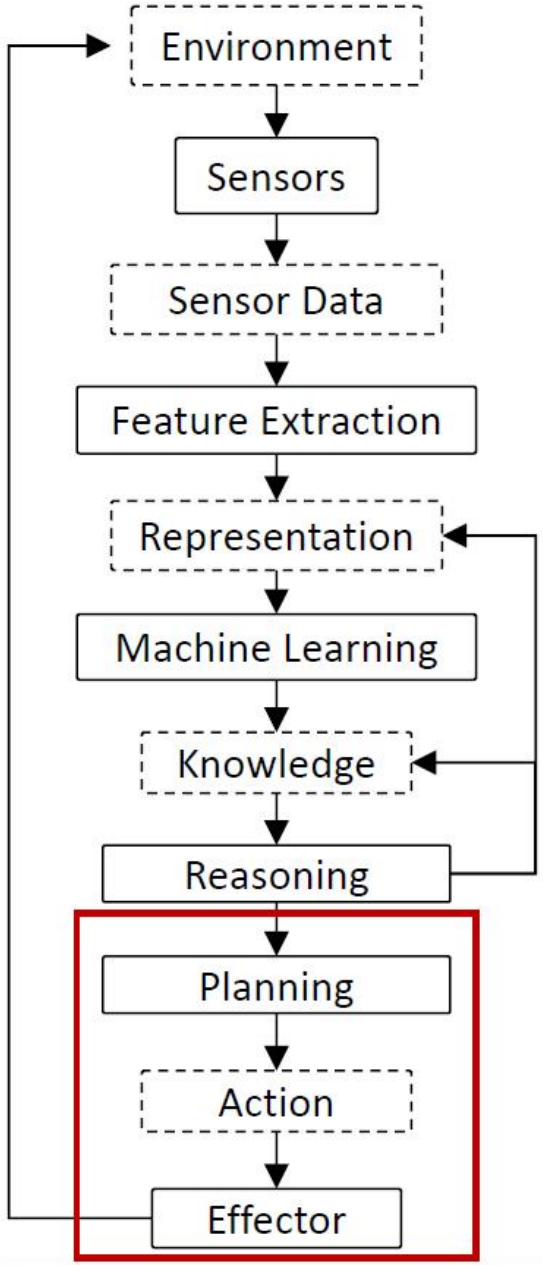


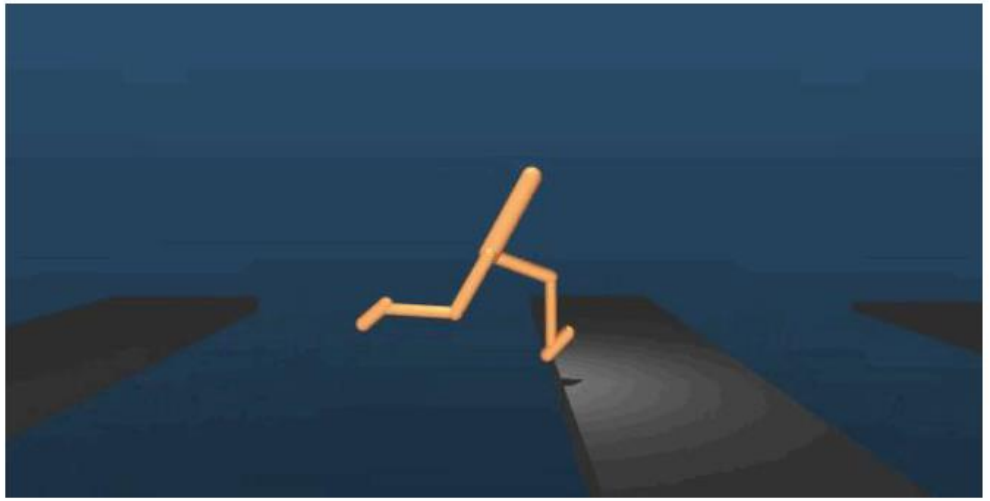
Image Recognition:
If it looks like a duck

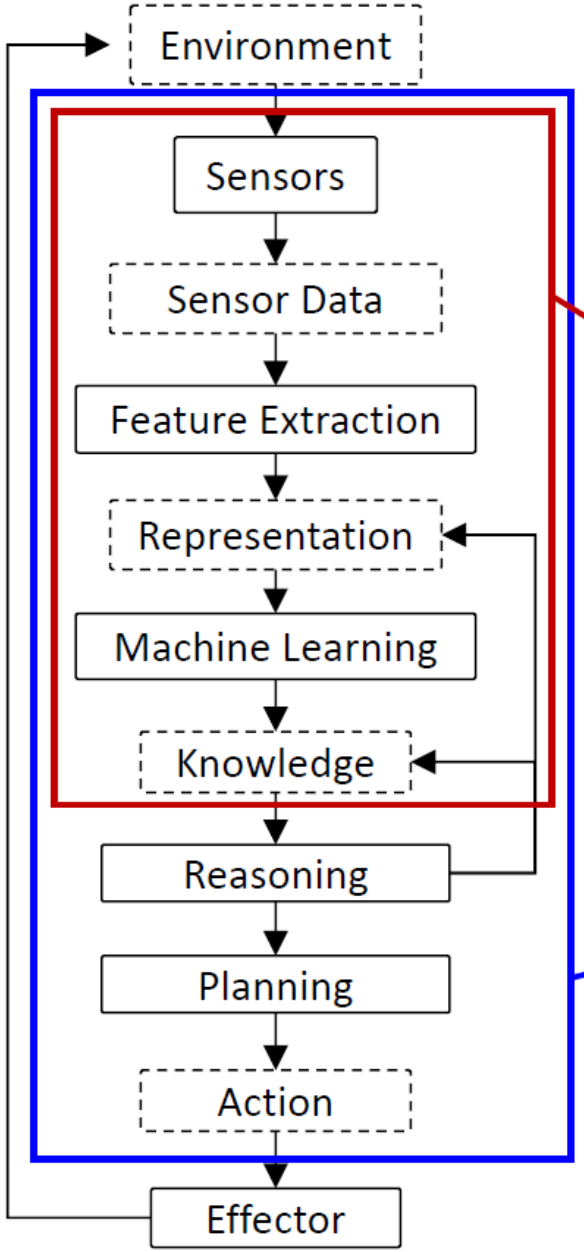
Audio Recognition:
Quacks like a duck



Activity Recognition:
Swims like a duck


Environment
Sensors
Sensor Data
Feature Extraction
Representation
Machine Learning
Knowledge
Reasoning
Planning
Action
Effector





The promise of
Deep Learning

The promise of
Deep Reinforcement Learning

Q-LEARNING AND DEEP Q-LEARNING

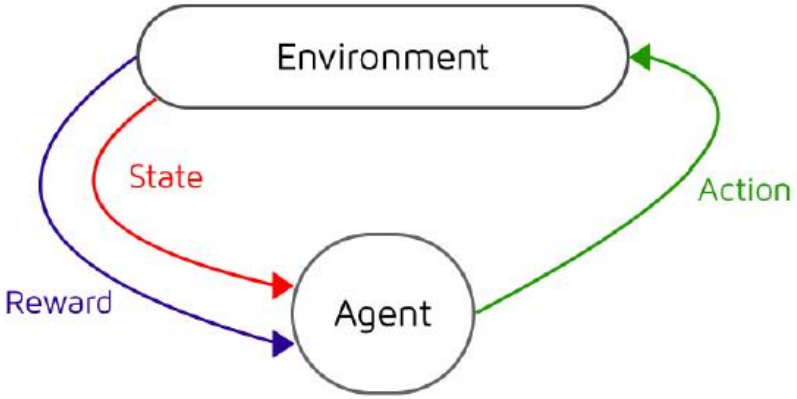
Reinforcement Learning Framework

At each step, the agent:

- Executes action
- Observe new state
- Receive reward

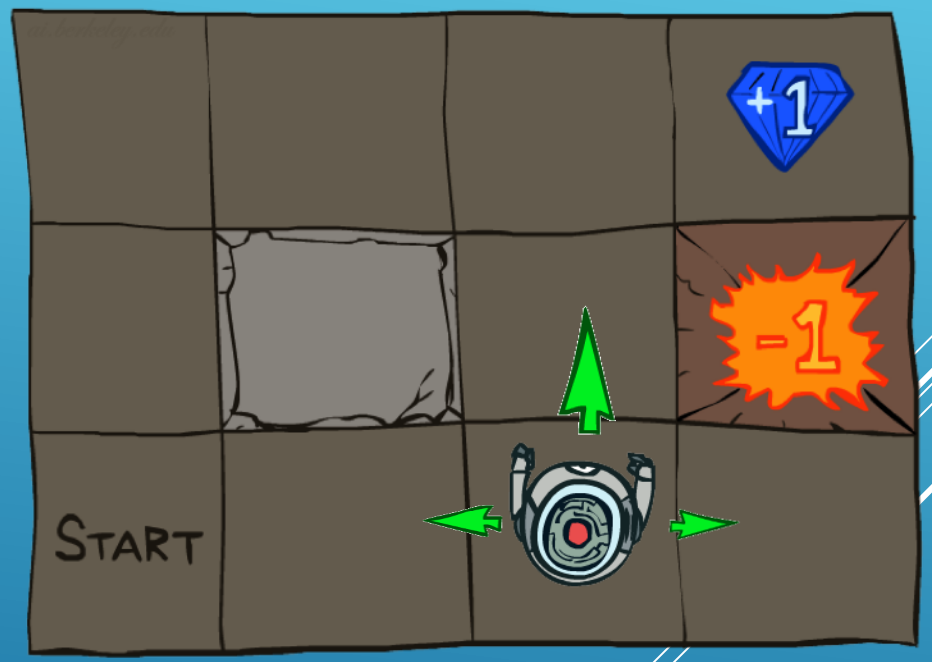
Open Questions:

- What cannot be modeled in this way?
- What are the challenges of learning in this framework?



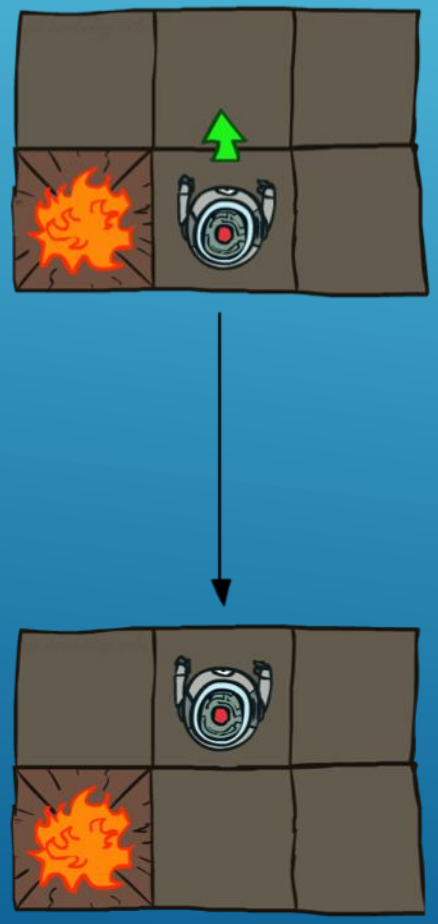
GRID WORLD

- A maze-like problem
  - The agent lives in a grid
  - Walls block the agent's path
- Noisy movement: actions do not always go as planned
  - 80% of the time, the action North takes the agent North (if there is no wall there)
  - 10% of the time, North takes the agent West; 10% East
  - If there is a wall in the direction the agent would have been taken, the agent stays put
- The agent receives rewards each time step
  - Small "living" reward each step (can be negative)
  - Big rewards come at the end (good or bad)
- Goal: maximize sum of rewards

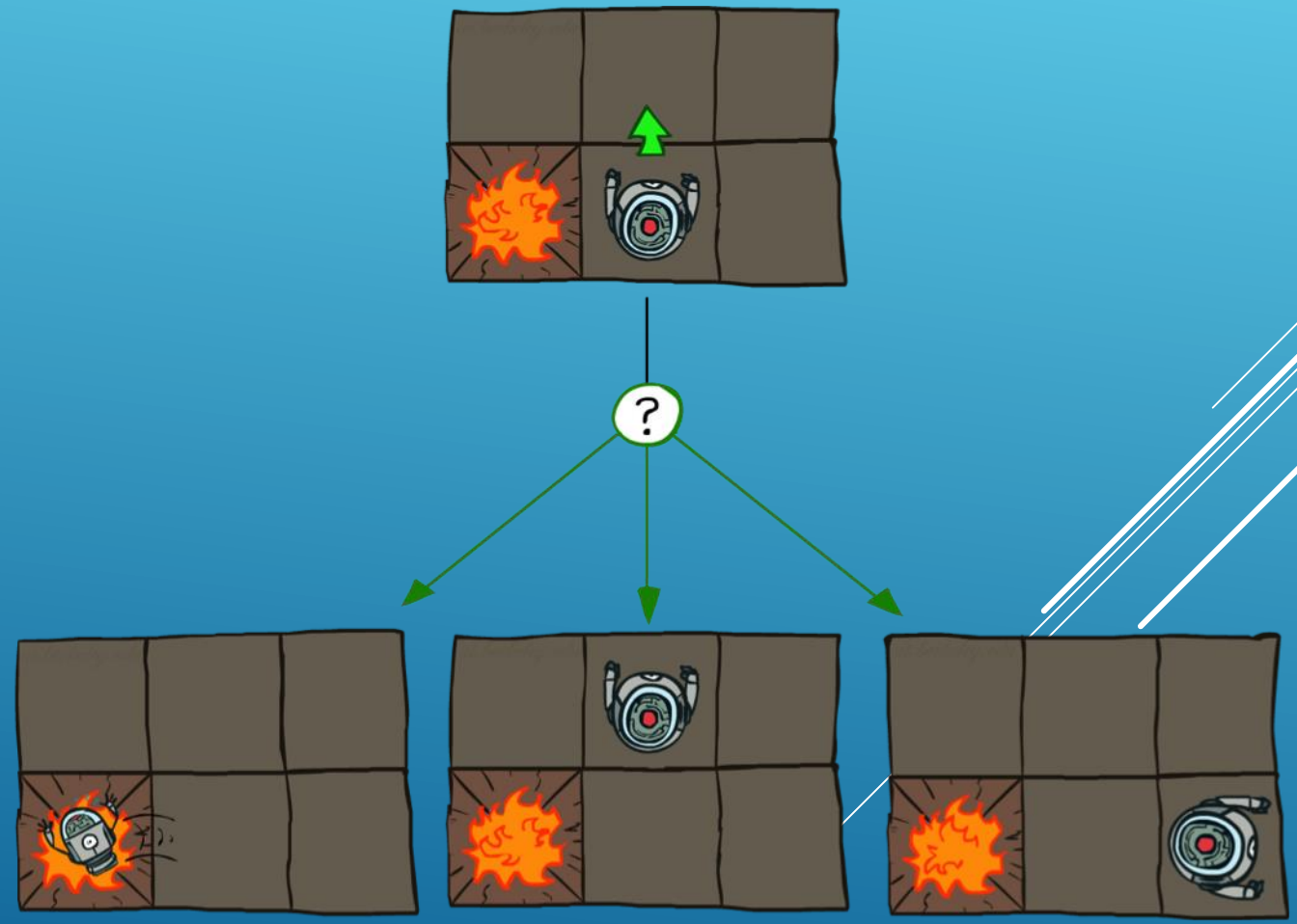


GRID WORLD ACTIONS

Deterministic Grid World



Stochastic Grid World



Q-LEARNING UPDATE RULE

- On transitioning from state *s* to state *s'* on action *a*, and receiving reward *r*, update:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q_k(s',a')\right]$$

- α is the **learning rate**
  - A large α results in quicker learning but may not converge.
  - α is often decreased as learning goes on.
- γ is the **discount rate** i.e., discounts future rewards

Q-LEARNING ALGORITHM

For each state *s* and action *a*:

$$Q(s,a) \leftarrow 0$$

Begin in state s:

Repeat:

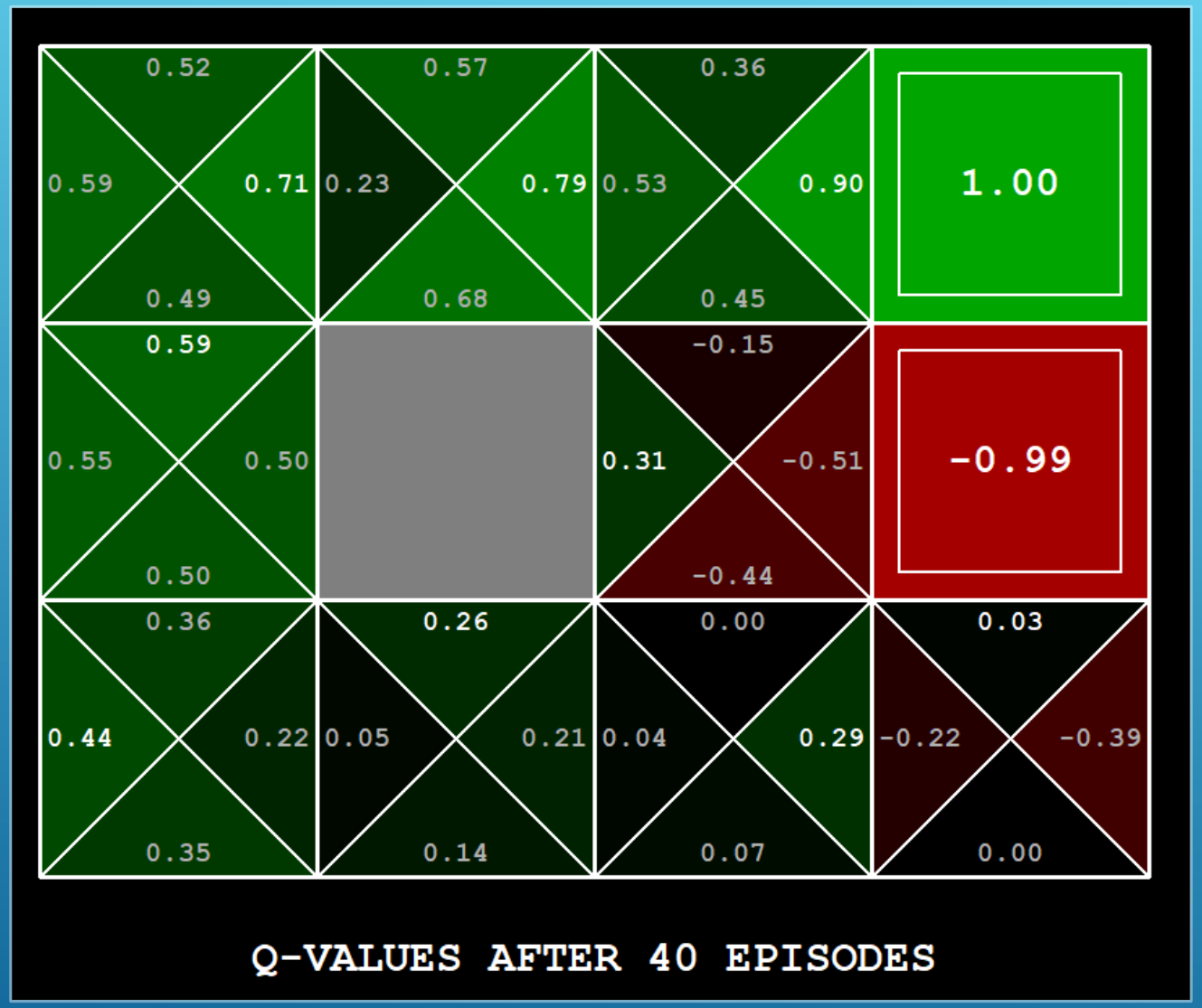
For all actions associated with state *s*,
→ **CHOOSE ACTION *a*** ← based on the Q values for state *s*

Receive reward *r* and transition to s'

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + (\alpha)\left[r + \gamma \max_{a'} Q(s',a')\right]$$

$$s \leftarrow s'$$

Q-LEARNING EXAMPLE: GRIDWORLD



Q FUNCTION: Q(S,A)

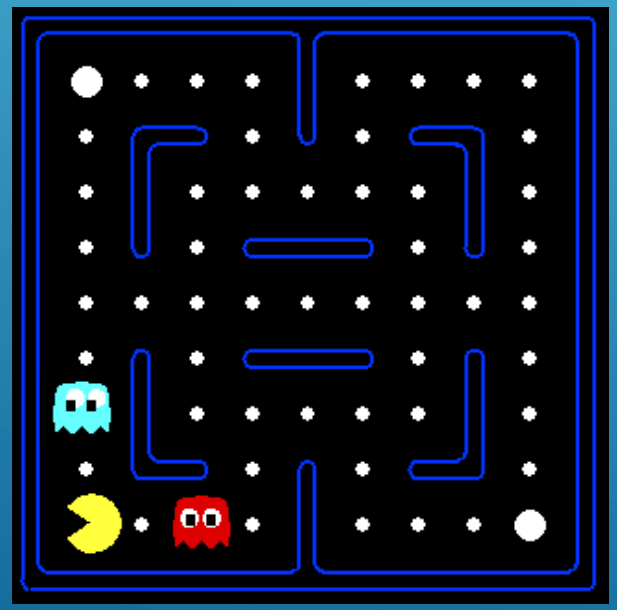
Action A

State S

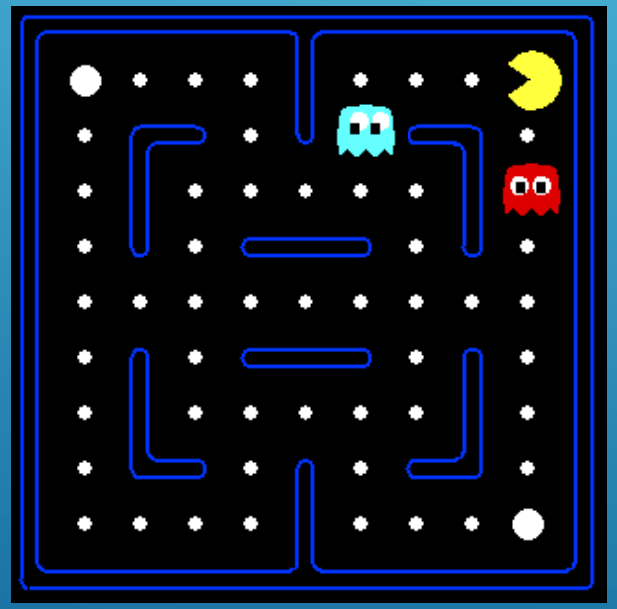
	A1	A2	A3	A4
S1	+1	+2	-1	0
S2	+2	0	+1	-2
S3	-1	+1	0	-2
S4	-2	0	+1	+1

EXAMPLE: PACMAN

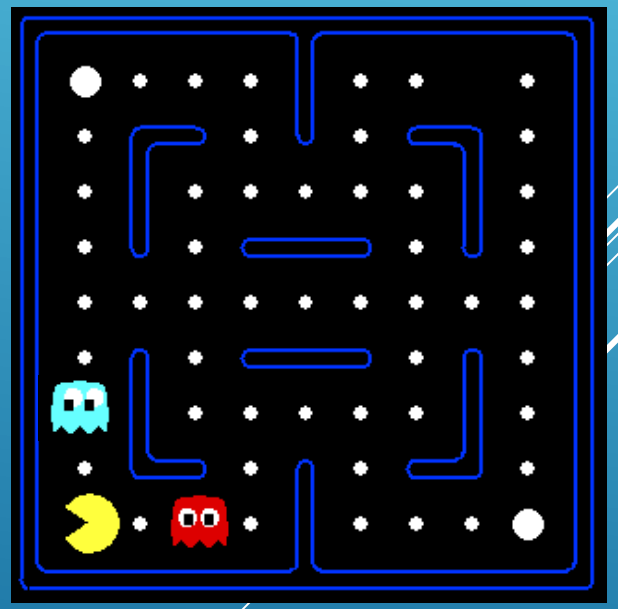
Let's say we discover through experience that this state is bad:



In naïve q-learning, we know nothing about this state:



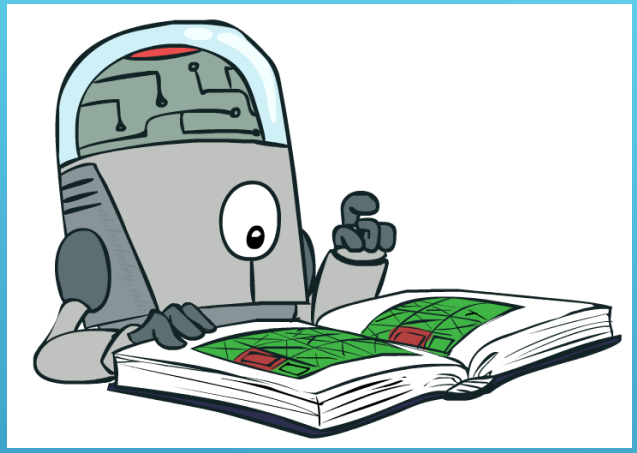
Or even this one!



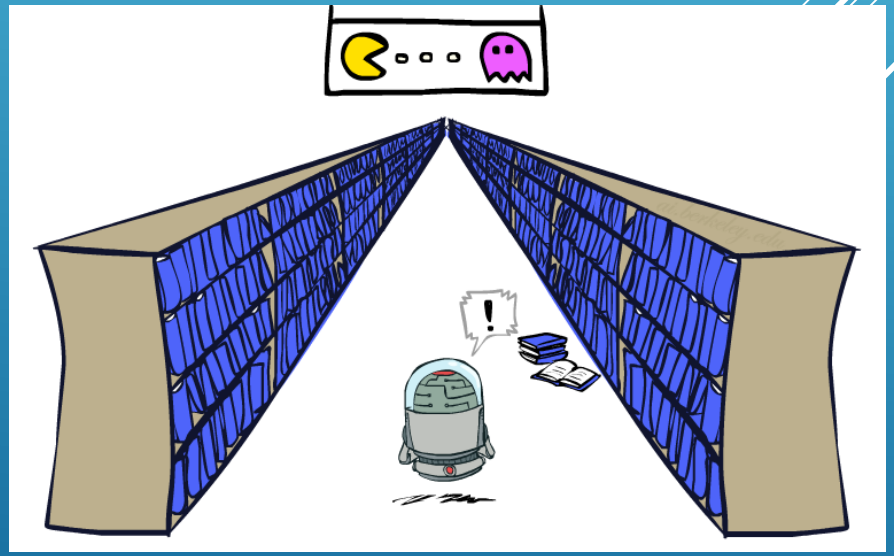
Credit: CS188 – Intro to AI, UC Berkeley, ai.berkely.edu

GENERALIZING ACROSS STATES

- In realistic situations, we cannot possibly learn about every single state!
  - Too many states to visit them all in training
  - Too many states to hold the q-tables in memory



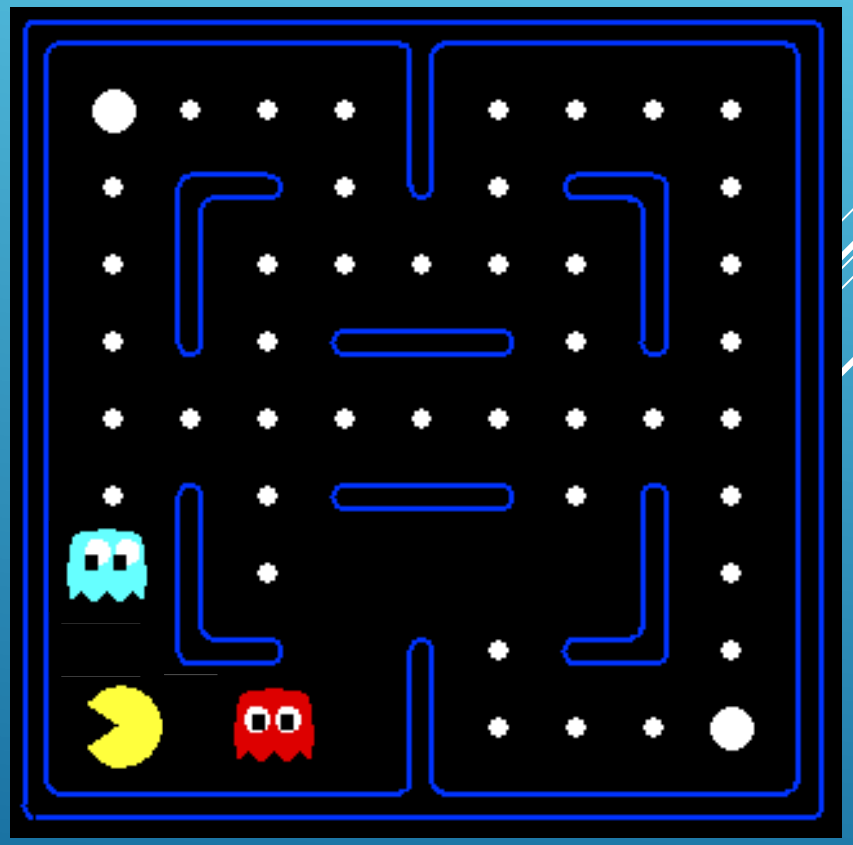
- Instead, we want to generalize:
  - Learn about some small number of training states from experience
  - Generalize that experience to new, similar situations



APPROXIMATE Q-LEARNING

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$$

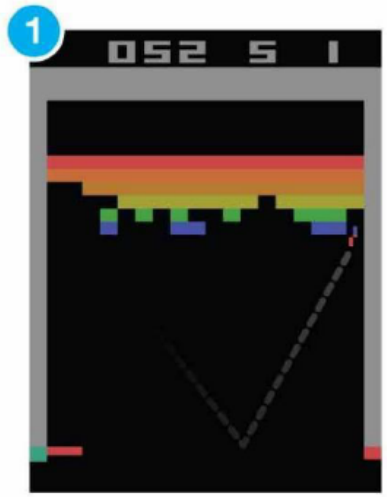
- Describe a state using a vector of features (properties)
  - Features are functions from states to real numbers (often 0/1) that capture important properties of the state
  - Example features:
    - Distance to closest ghost
    - Distance to closest dot
    - Number of ghosts
    - 1 / (dist to dot)^2
    - Is Pacman in a tunnel? (0/1)
    - …… etc.



Credit: CS188 – Intro to AI, UC Berkeley, ai.berkely.edu

Q-Learning: Representation Matters

- In practice, Value Iteration is impractical
  - Very limited states/actions
  - Cannot generalize to unobserved states



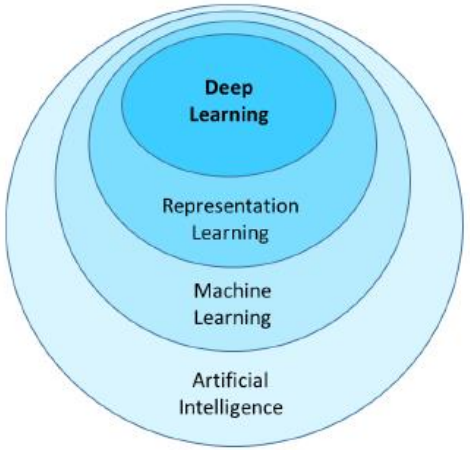
- Think about the **Breakout** game
  - State: screen pixels
    - Image size: 84×84 (resized)
    - Consecutive **4** images
    - Grayscale with **256** gray levels

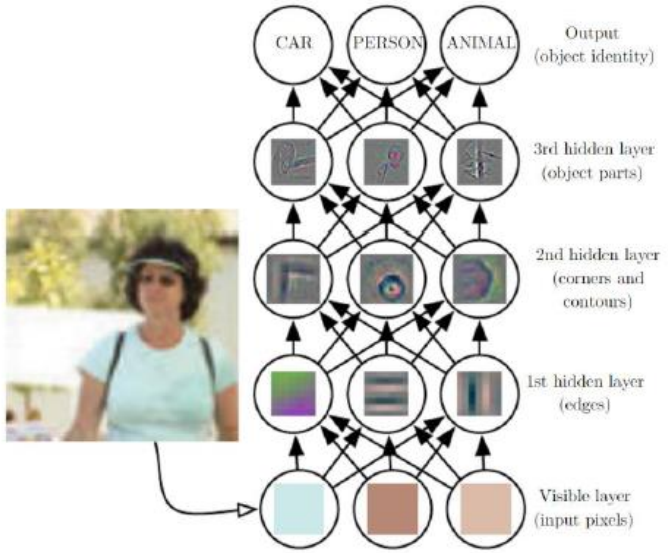
$256^{84 \times 84 \times 4}$ rows in the Q-table!

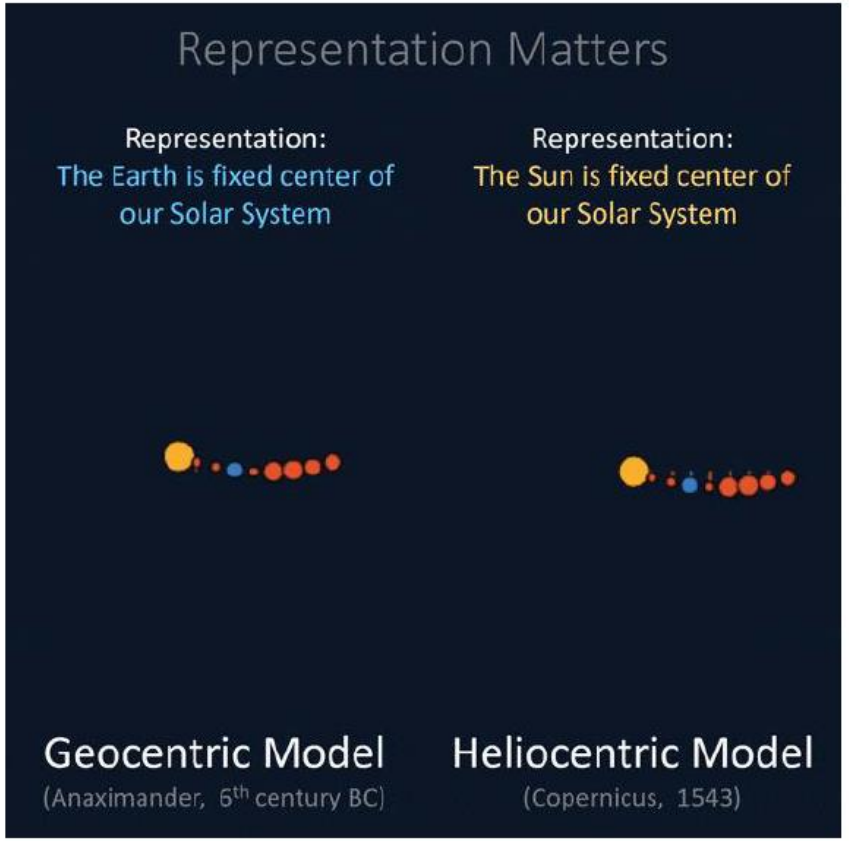
$= 10^{69,970} >> 10^{82}$ atoms in the universe

References: [83, 84]

Deep RL = RL + Neural Networks

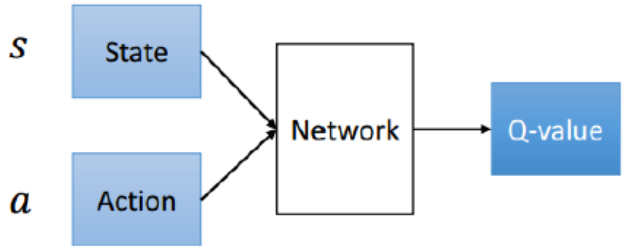




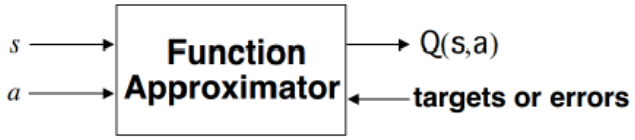


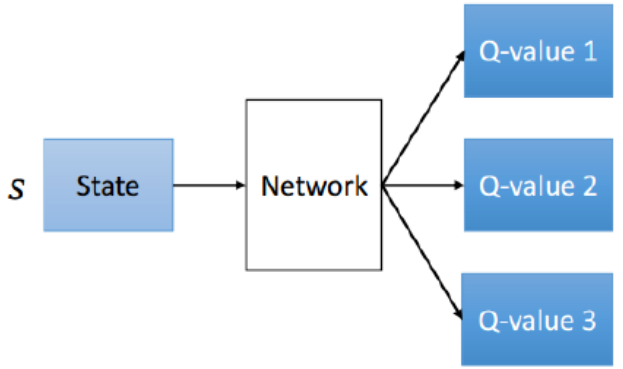
DQN: Deep Q-Learning

Use a neural network to approximate the Q-function:



$$Q(s, a; \boldsymbol{\theta}) \approx Q^*(s, a)$$





TRAINING REAL-WORLD REINFORCEMENT LEARNING AGENTS

The Challenge for RL in Real-World Applications

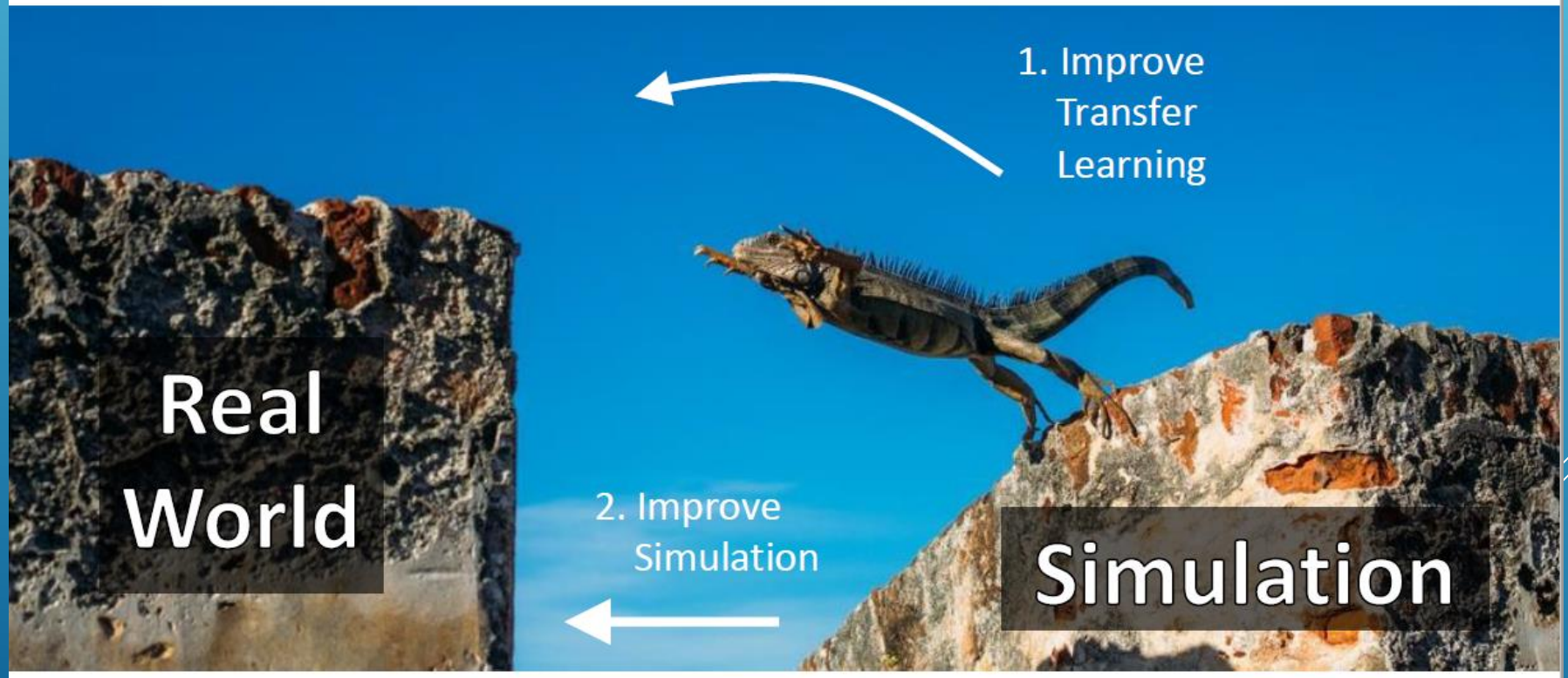
Reminder:

Supervised learning:
teach by example

Reinforcement learning:
teach by experience

Open **Challenges**. Two Options:

1. Real world observation + one-shot trial & error
2. Realistic simulation + transfer learning



WANT THESE SLIDES?

Email Scott O'Hara at seohara@gmail.com and ask me for them.