Deep Q-Learning

Industrial AI Lab.
Prof. Seungchul Lee

Q-Learning

- Value iteration algorithm: use Bellman equation as an iterative update

$$Q_{k+1}(s,a) \;\leftarrow\; \mathbb{E}_{s' \sim P(s'|s,a)}\left[R(s) + \gamma \max_{a} Q_k\left(s', a'\right)\right]$$

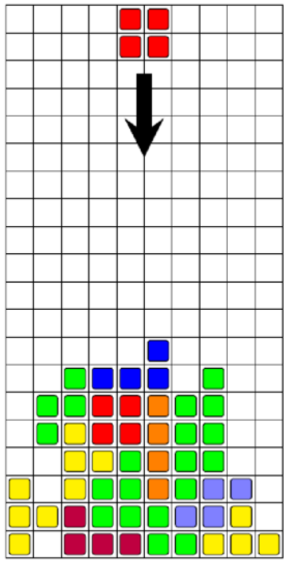
- Q_k will converge to Q_* as goes to ∞

What is the Problem with Q-Learning?

- Not scalable. Must compute tabular $Q(s, a)$ for every state-action pair.
- If state is for instance current game state pixels, computationally infeasible to compute for entire state space !
  - too many states to visit them all in training
  - too many states to hold the Q-table in memory

- States in Tetris: naive board configuration + shape of the falling piece $\sim 10^{60}$ states !!!

- Solution: use a function approximator to estimate $Q(s, a)$
  - learn about some small number of training states from experience
  - generalize that experience to new, similar situations



Tetris

Approximate Q-Learning

- Instead of a table, we have a parameterized Q function: $Q_\omega(s, a) \approx Q_*(s, a)$

- can be a linear function in features

$$Q_\omega(s,a) = \omega_0 f_0(s,a) + \omega_1 f_1(s,a) + \cdots + \omega_n f_n(s,a)$$

- or a complicated neural network → (Deep Q-learning Network)

$$\text{target}\,(s') = R(s) + \gamma \max_{a'} Q_\omega(s', a')$$

$$Q_{k+1}(s,a) \;\leftarrow\; Q_k(s,a) + \alpha\, \underbrace{\left(\text{target}\,(s') - Q_k(s,a)\right)}_{\text{difference}}$$

Approximate Q-Learning

- Difference

$$\text{difference} = \left[R(s) + \gamma \max_{a'} Q_\omega(s', a')\right] - Q_\omega(s, a)$$

- loss function by mean-squared error in Q-value

$$\ell(\omega) = \left[\frac{1}{2}\left(\text{target}\,(s') - Q_\omega(s, a)\right)^2\right]$$

- GD update:

$$\begin{aligned} \omega_{k+1} &\leftarrow \omega_k - \alpha \, \nabla_\omega \ell(\omega) \qquad\qquad \Longrightarrow \text{ Deep Learning Framework} \\ &\leftarrow \omega_k - \alpha \, \nabla_\omega \left[\frac{1}{2}\left(\text{target}\,(s') - Q_\omega(s, a)\right)^2\right] \\ &\leftarrow \omega_k + \alpha \, \left[\left(\text{target}\,(s') - Q_\omega(s, a)\right)\right] \nabla_\omega Q_\omega(s, a) \end{aligned}$$

Linea Function Approximator

$$Q_\omega(s,a) = \omega_0 f_0(s,a) + \omega_1 f_1(s,a) + \cdots + \omega_n f_n(s,a)$$

- Exact Q

$$Q(s,a) \leftarrow Q(s,a) + \alpha\,[\text{difference}]$$

- Approximate Q

$$\omega_i \leftarrow \omega_i + \alpha\,[\text{difference}]\, f_i(s,a)$$

Example: Linear Regression

$$\ell = \frac{1}{2}\sum_i (y_i - \hat{y}_i)^2$$

- Difference

$$y - \hat{y} = y - (\omega_0 f_0(x) + \omega_1 f_1(x) + \cdots + \omega_n f_n(x)) = \underbrace{y}_{\text{target}} - \underbrace{\sum_i \omega_i f_i(x)}_{\text{prediction}}$$

- For one point

$$\frac{\partial \ell(\omega)}{\partial \omega_k} = \left(y - \sum_i \omega_i f_i(x)\right) f_k(x)$$

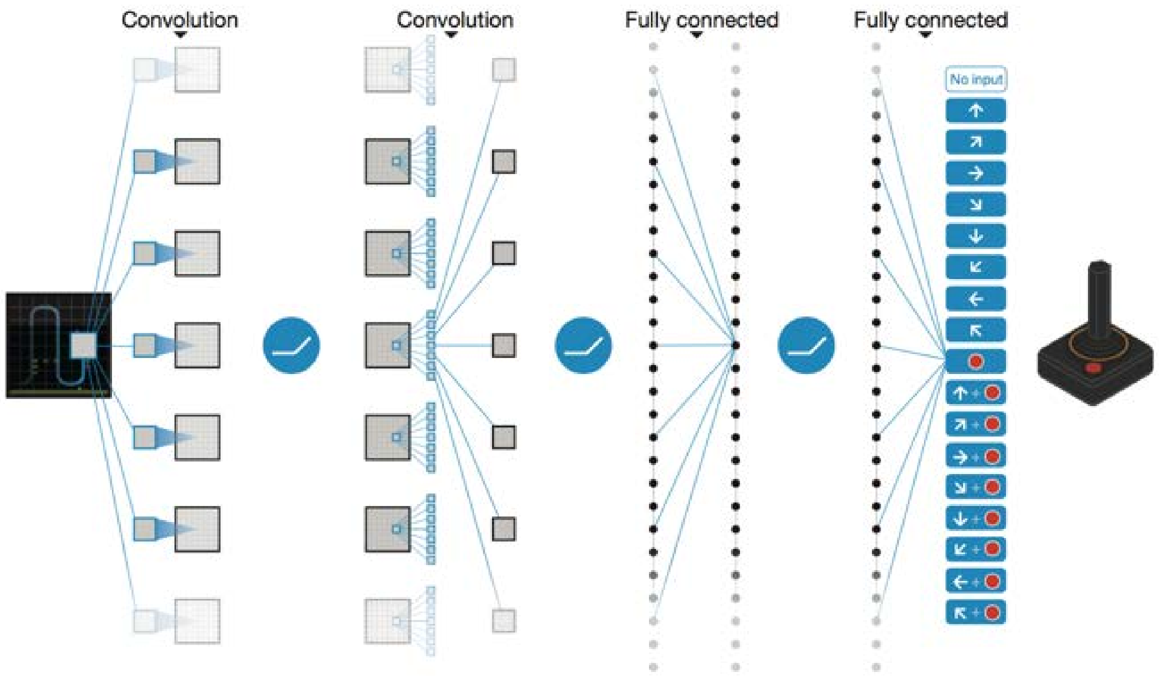
$$\omega_k \leftarrow \omega_i + \alpha \left(y - \sum_i \omega_i f_i(x)\right) f_k(x)$$

$$\omega_k \leftarrow \omega_i + \alpha\ [\text{difference}]\ f_k(x)$$

Deep Q-Networks (DQN)

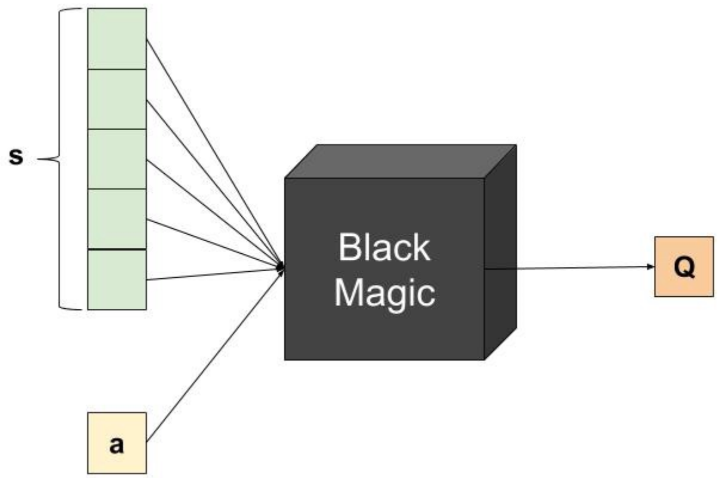
- DQN method (Q-learning with deep network as Q function approximator) became famous in 2013 for learning to play a wide variety of Atari games better than humans.

- Deep Mind, 2015
  - Used a deep learning network to represent Q

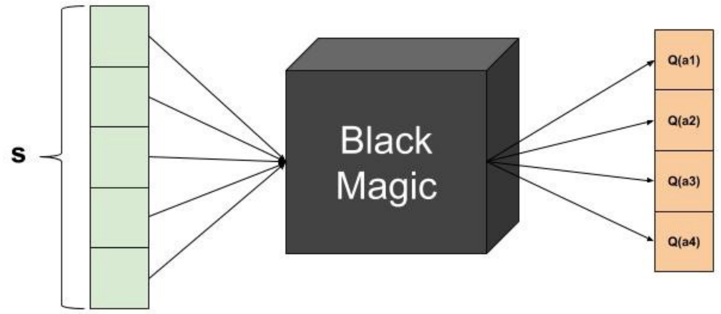


Deep Q-Networks (DQN)

- (Tabular) Q-Learning like DQN, but inefficient

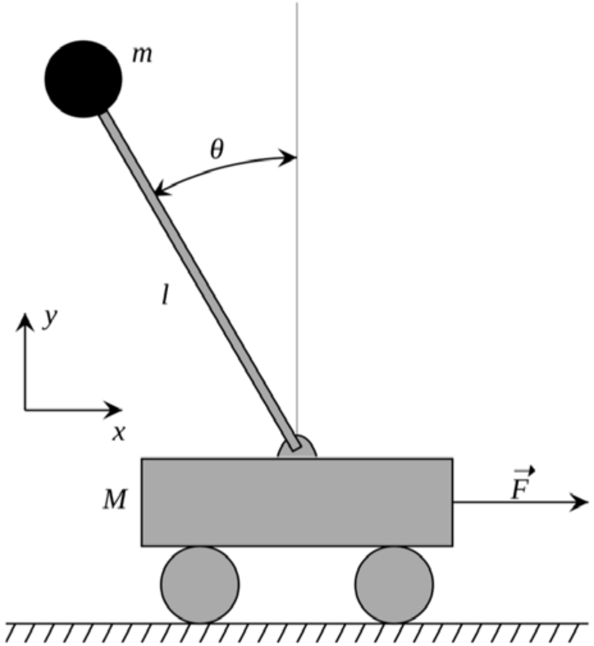


- Better DQN



DQN with Gym Environment

- Cart Pole Problem
  - Objective:
    - Balance a pole on top of a movable cart
  - State:
    - position, horizontal velocity, angle, angular speed
  - Action:
    - horizontal force applied on the cart
  - Reward:
    - 1 at each time step if the pole is upright



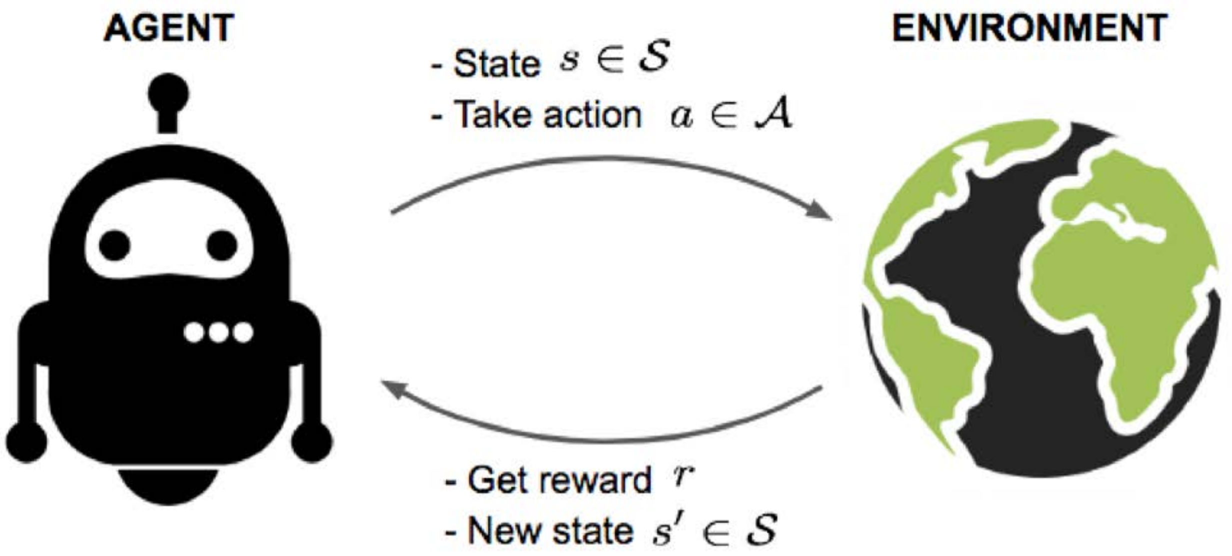
Nature of Learning

- We learn from past experiences
  - She has no explicit teacher but does have direct interaction to the environment

- Positive compliments vs. negative criticism

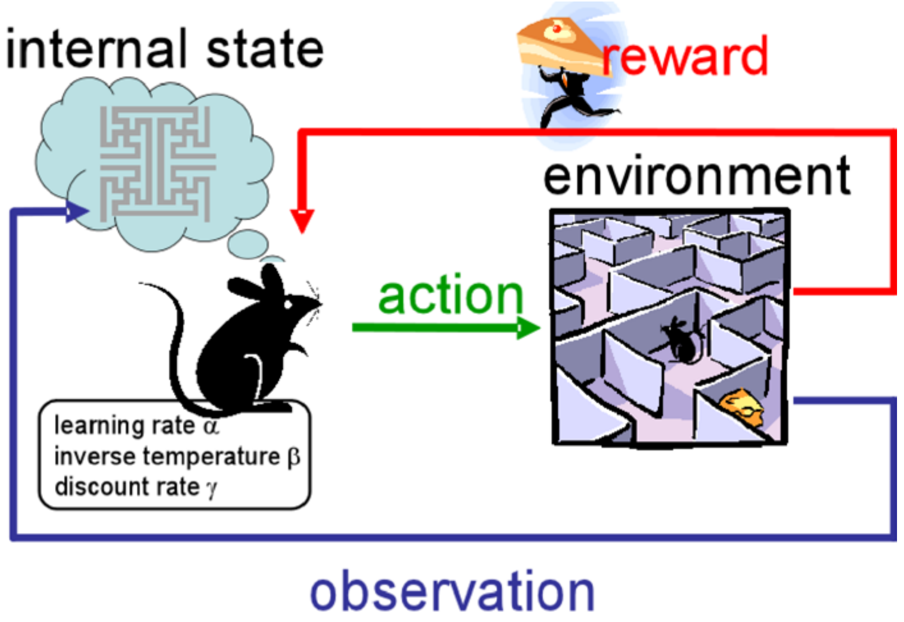
What is Reinforcement Learning?

- Computational approach to learning from interaction
  - Learn to make good sequence of decisions
  - No supervision
  - Feedback is delayed
  - Actions affect the subsequent future rewards
- The key challenge is to learn to make good decision under uncertainty



Fundamental Terminology in RL

- Markov Decision Process (MDP)
  - State, action
  - State transition probability, reward function, discount factor
- Policy, value, model
- Planning vs. learning
- Predictions vs. control
- Exploration vs. exploitation



From MDP To Reinforcement Learning

- You should take good actions to get rewards, but in order to know which actions are good, we need to explore and try different actions.

- Markov decision process (offline)
  - Have mental model of how the world works.
  - Find policy to collect maximum rewards.

- Reinforcement learning (online)
  - Don't know how the world works.
  - Perform actions in the world to find out and collect rewards.

Deep Reinforcement Learning

- Playing Atari [Google DeepMind, 2013]:

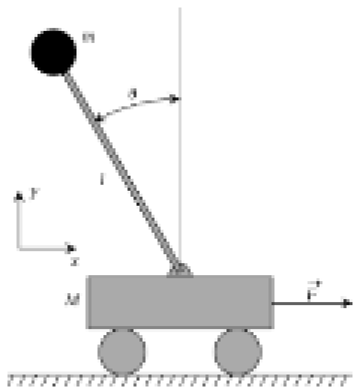
- Just use a neural network for $\hat{Q}_{\text{opt}}(s, a)$
- Last 4 frames (images) → 3-layer NN → keystroke
- ϵ-greedy, train over 10M frames with 1M replay memory
- https://www.youtube.com/watch?v=V1eYniJ0Rnk

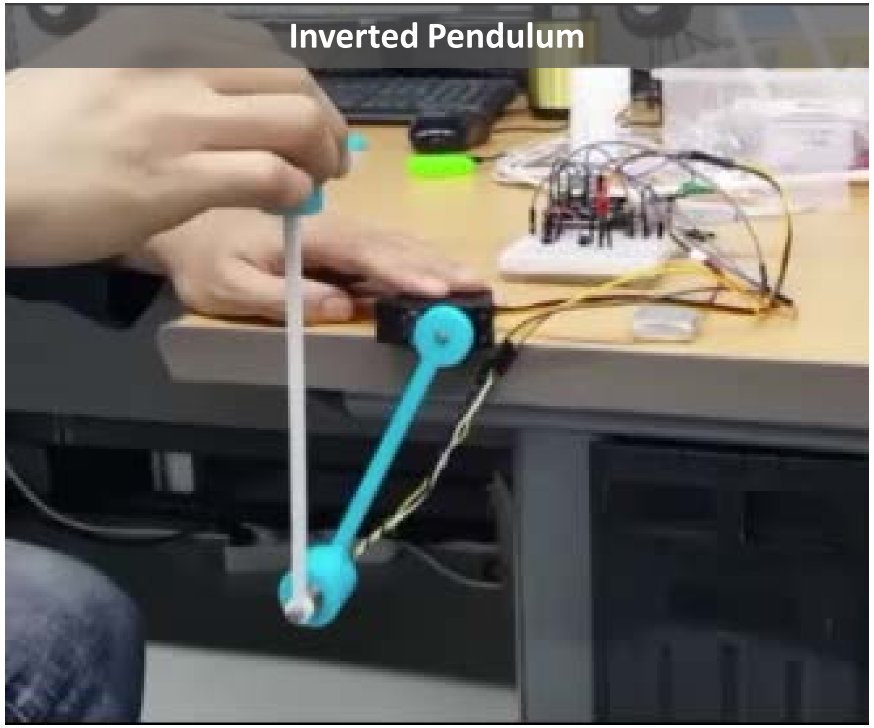
AlphaGo

- Supervised learning: on human games
- Reinforcement learning: on self-play games
- Evaluation function: convolutional neural network (value network)
- Policy: convolutional neural network (policy network)
- Monte Carlo Tree Search: search / lookahead

Control Inverted Pendulum

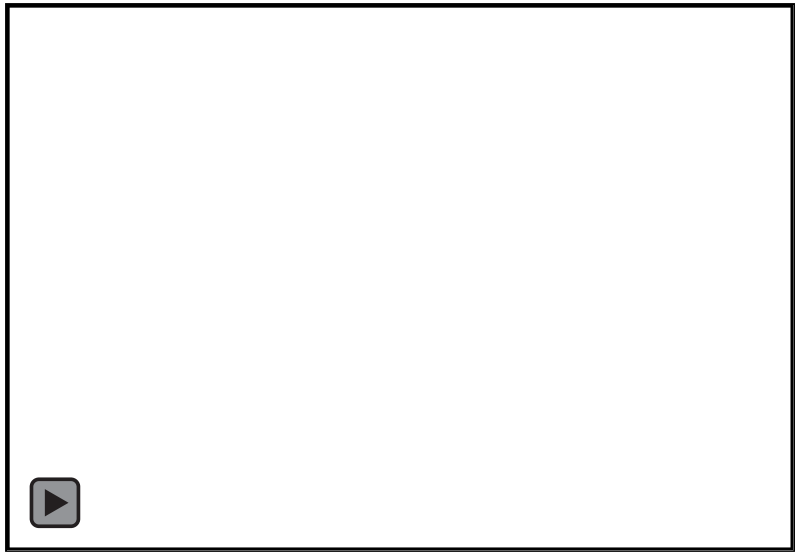
- From open-loop to closed-loop systems

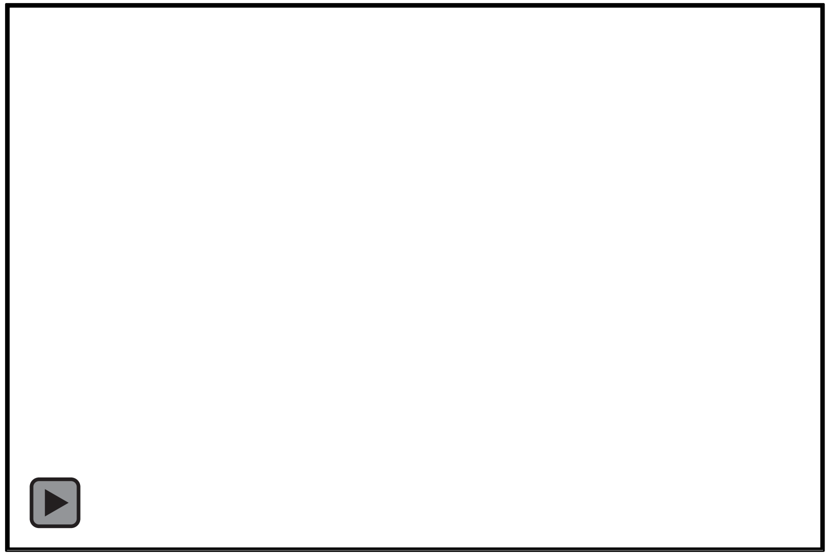




Reinforcement Learning

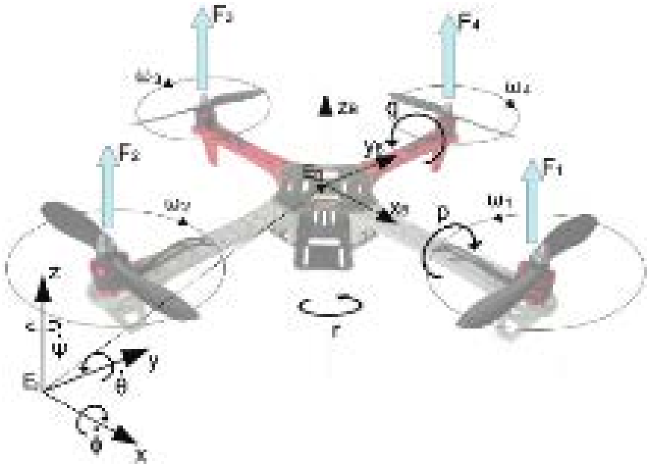
- Software-in-the-loop

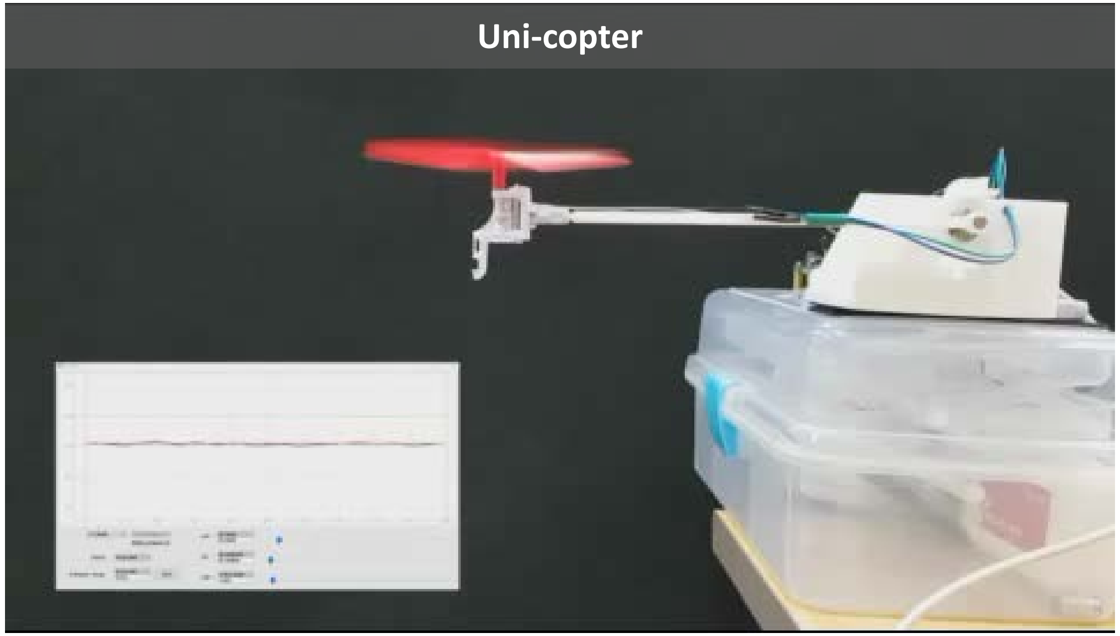




Control Uni-copter

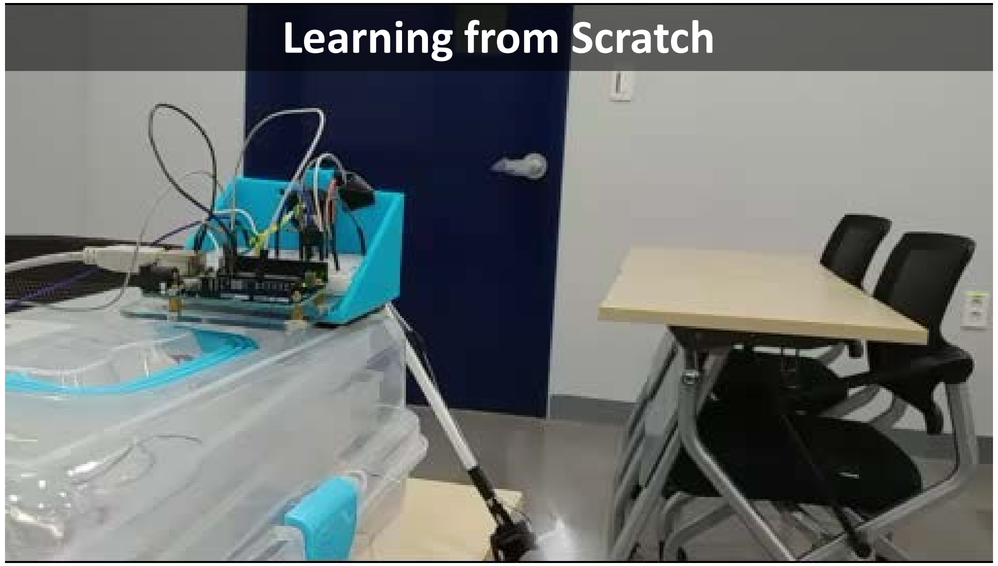
- From open-loop to closed-loop systems

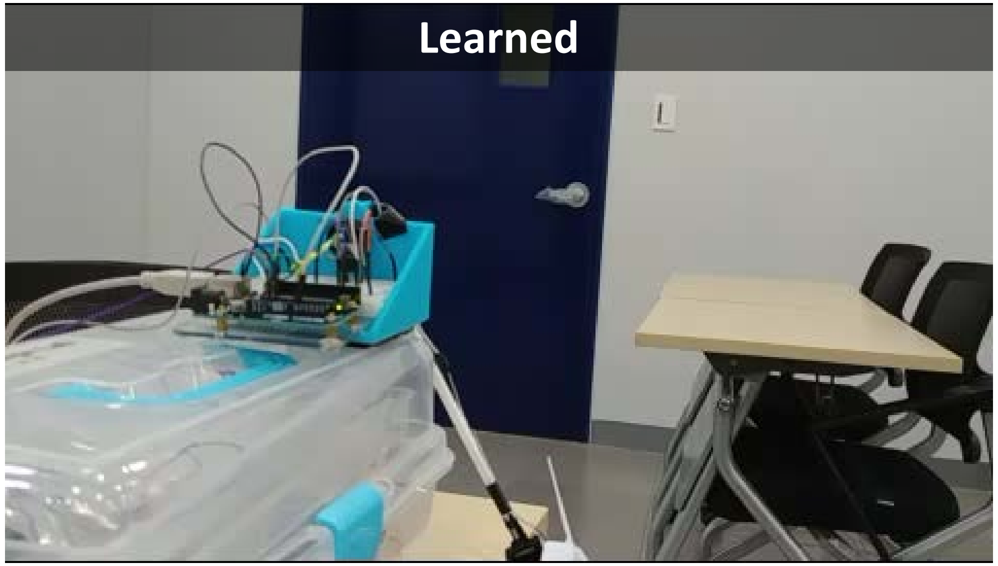




Reinforcement Learning

- Hardware-in-the-loop





Reinforcement Learning

- Learned knowledge can be transferred