Faster ReinforcementLearning

via Transfer

John Schulman

Wonseok Jung

HUMAN. MACHINE.
EXPERIENCE TOGETHER

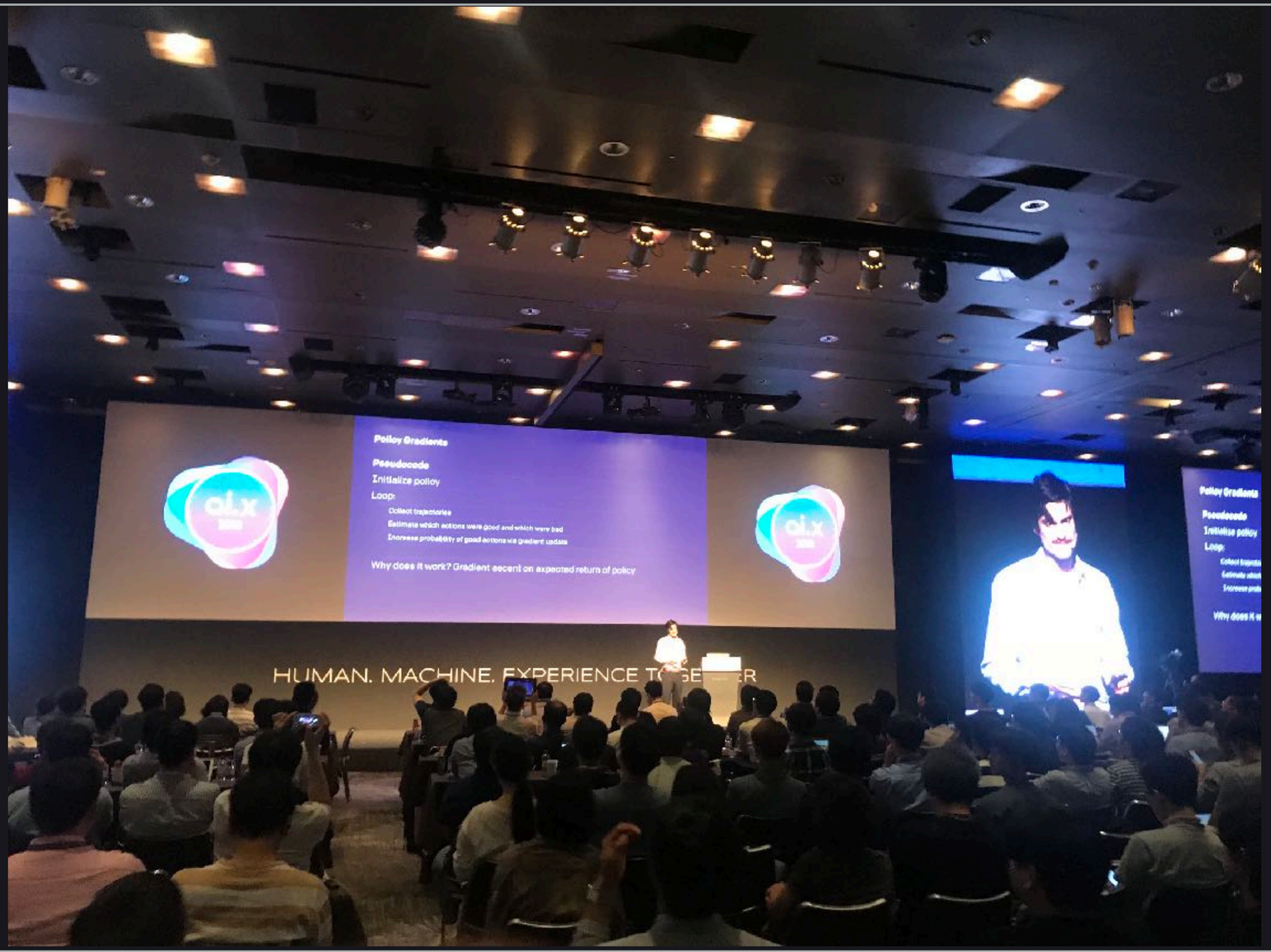
- 2018.09.06 09:00~17:00
- WALKERHILL HOTEL VISTA HALL

FASTER REINFORCEMENT LEARNING VIA TRANSFER

Faster Reinforcement Learning via Transfer

John Schulman Research Scientist at Open AI

John is a research scientist at OpenAI and member of the founding team. He co-developed some of the most widely used reinforcement learning algorithms (TRPO and PPO) and software packages (OpenAI Gym and Baselines). He received a PhD in CS from UC Berkeley and a BS in Physics from Caltech. He was selected as one of MIT Technology Review's '35 Innovators Under 35' in 2018.


Policy Gradients
Pseudocode
Initialize policy
Loop:
Collect trajectories
Estimate which actions were good and which were bad
Increase probability of good actions via gradient update
Why does it work? Gradient ascent on expected return of policy
HUMAN. MACHINE. EXPERIENCE

OVERVIEW

1. Policy Gradients

   Success Stories

   Limitations

2. Meta Reinforcement Learning

3. Gym retro

TERMINOLOGY

Reinforcement Leaning :

Trial and error을 하며 Reward를 최대화 한다.

Deep R.L :

Neural network를 사용하여 RL algorithm을 represent한 것

Meta-Learning :

"Learning how to Learn" 어떠한 Learning에 관여하는 Task를 Master

TERMINOLOGY

Reinforcement Leaning :

Trial and error을 하며 Reward를 최대화 한다.

Deep R.L :

Neural network를 사용하여 RL algorithm을 represent한 것

Meta-Learning :

"Learning how to Learn" 어떠한 Learning에 관여하는 Task를 Master

TERMINOLOGY

Reinforcement Leaning :

Trial and error을 하며 Reward를 최대화 한다.

Deep R.L :

Neural network를 사용하여 RL algorithm을 represent한 것

Meta-Learning :

"Learning how to Learn" 어떠한 Learning에 관여하는 Task를 Master

TERMINOLOGY

Reinforcement Leaning :

Trial and error을 하며 Reward를 최대화 한다.

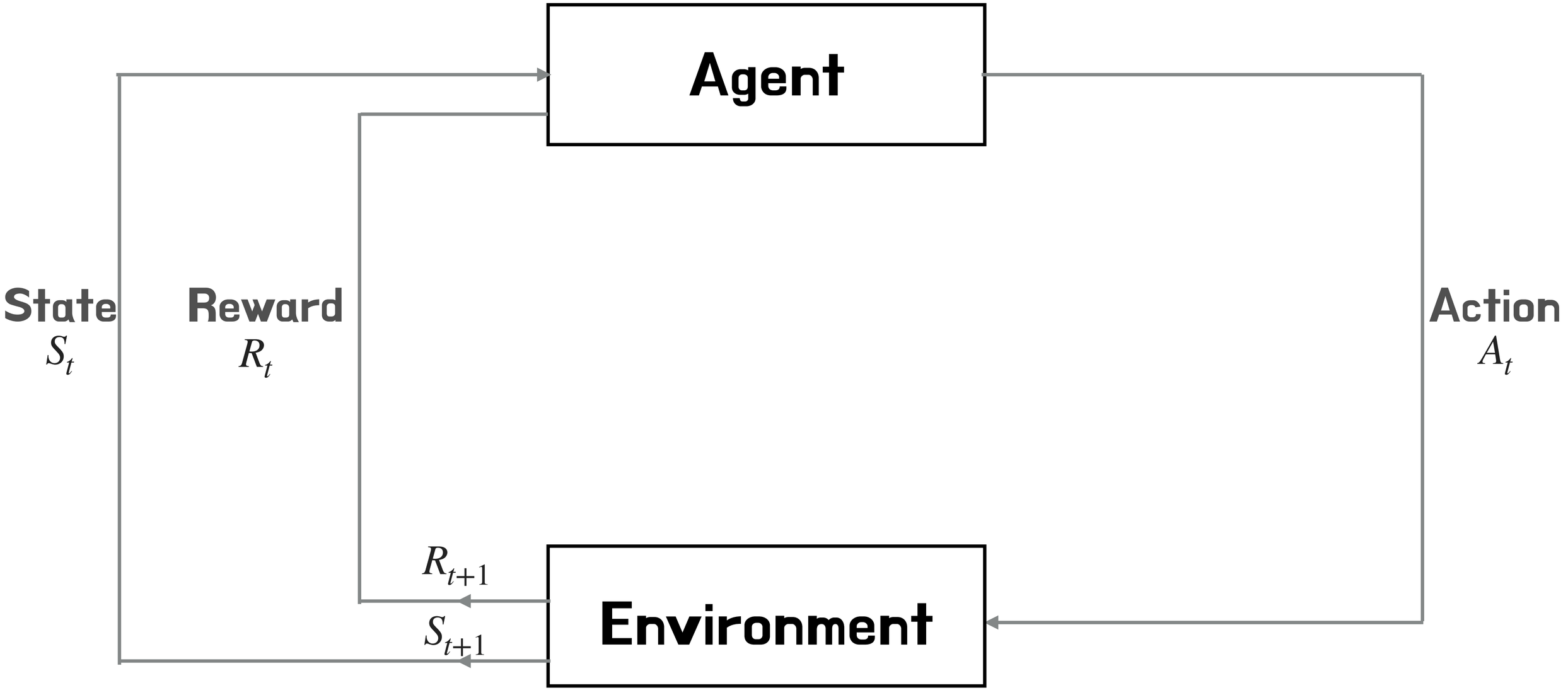
Deep R.L :

Neural network를 사용하여 RL algorithm을 represent한 것

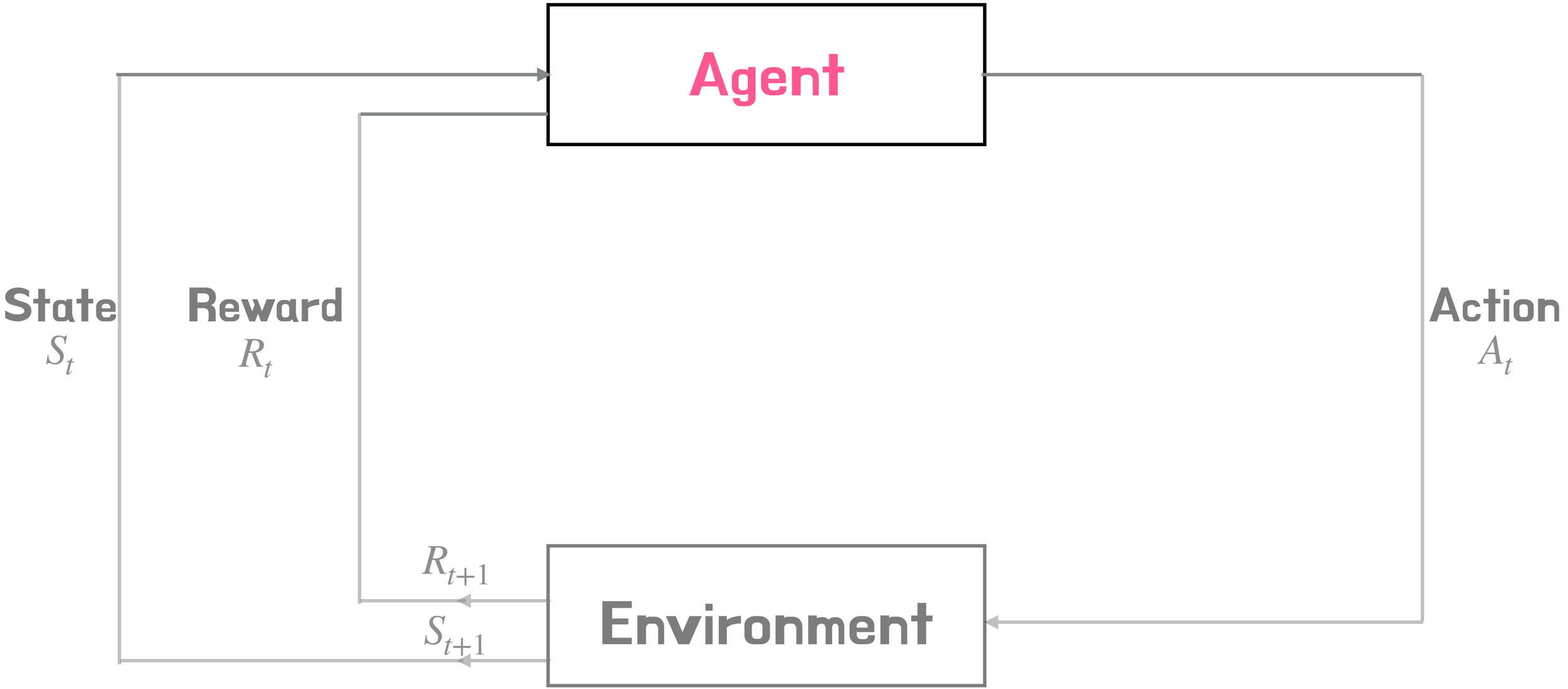
Meta-Learning :

“Learning how to Learn” 어떠한 Learning에 관여하는 Task를 Master

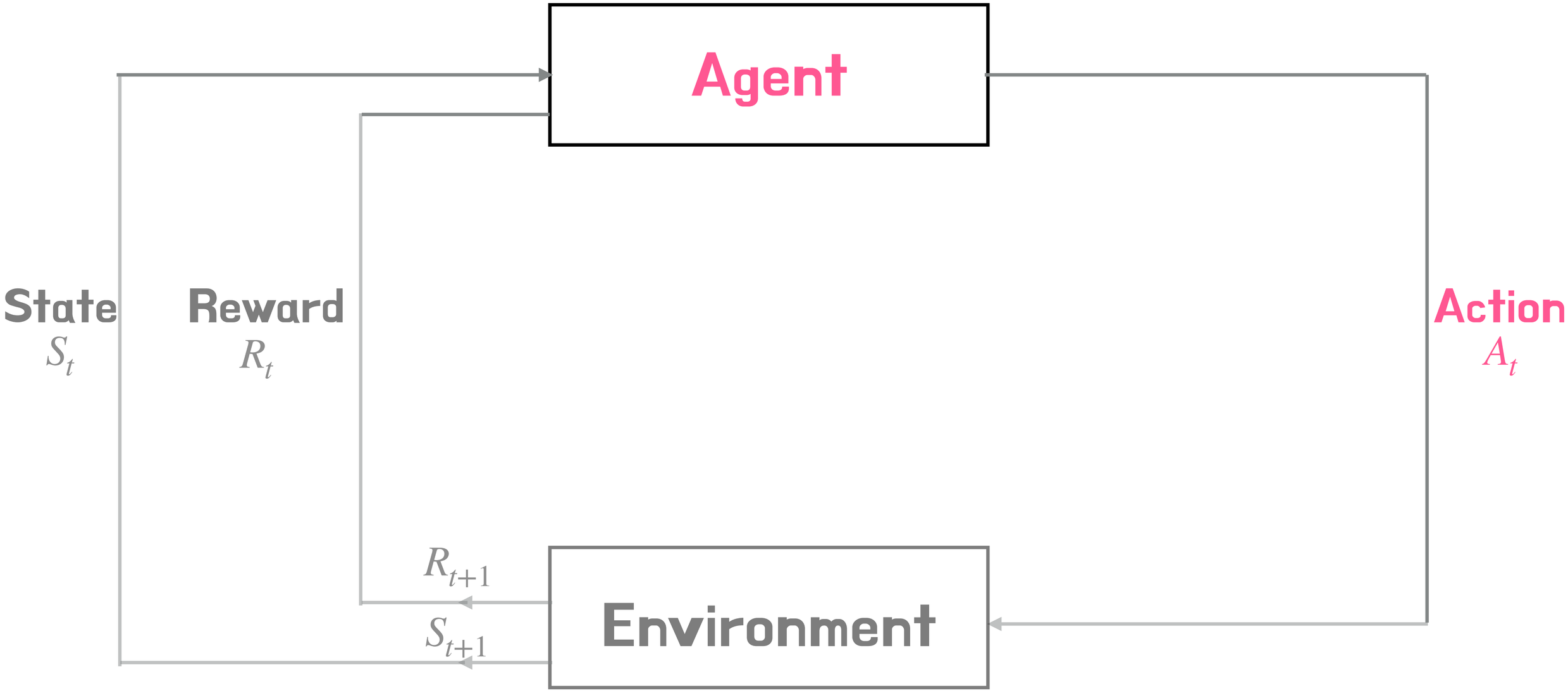
MARKOV DECISION PROCESS



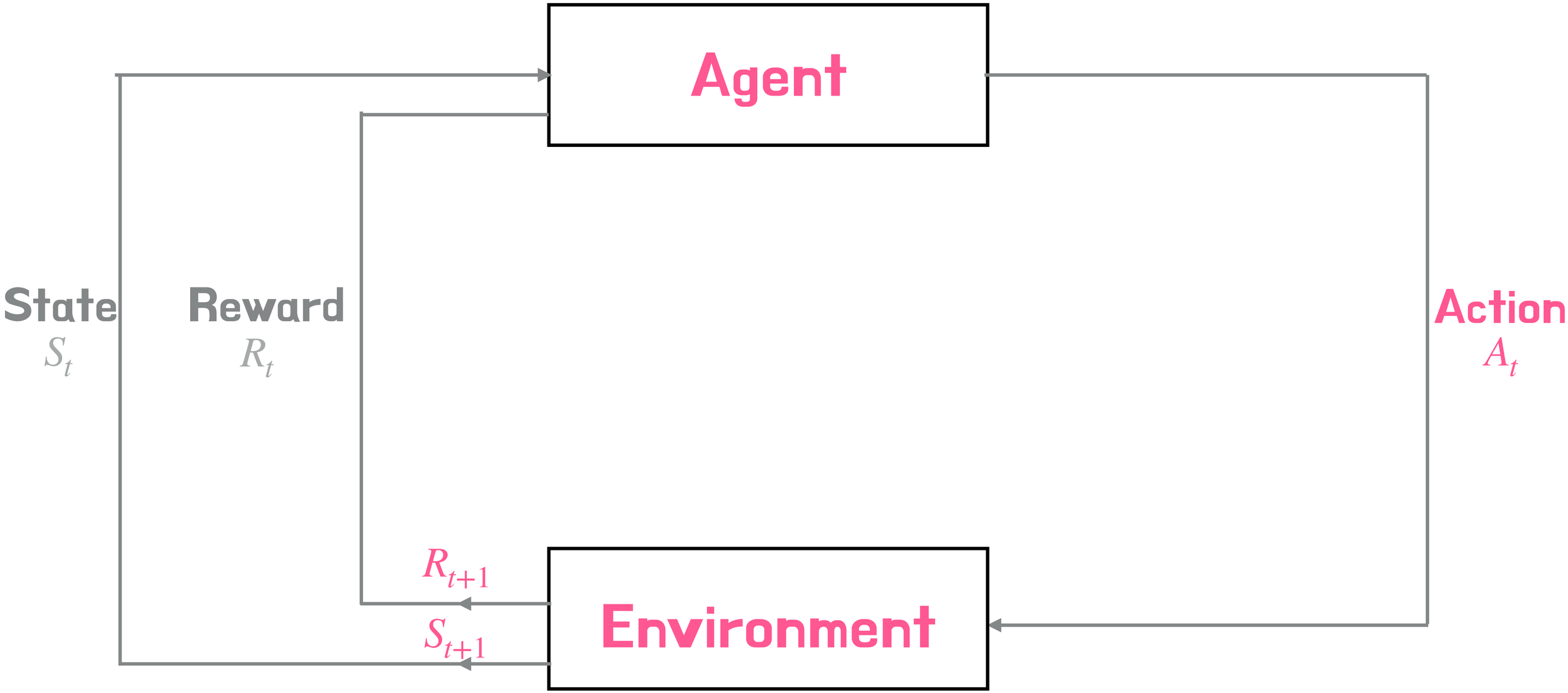
AGENT



ACTION



OBSERVATION, REWARD



TRAJECTORY

$$(S_t, A_t, R_{t+1}, S_{t+1}) \rightarrow (S_{t+1}, A_{t+1}, R_{t+2}, S_{t+2}) \rightarrow (S_{t+2}, A_{t+2}, R_{t+3}, S_{t+3})$$

RETURN

State-value

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right], \text{ for all } s \in \mathcal{S}$$

State-Action value

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right]$$

1. POLICY GRADIENTS

POLICY

Policy :

Observation에 의해 Action을 선택하는 함수

Policy Gradients method:

더 좋은 Policy를 찾기 위해서 Policy 자체를 최적화하는 강화학습 알고리즘

Policy gradients에 대해 좀 더 자세히 알고 싶다면

https://wonseokjung.github.io//reinforcementlearning/update/RL-PG_RE/

PSEUDO CODE

Pseudocode

Initialize policy

Loop:

- Collect trajectories
- Estimate which actions were good and which were bad
- Increase probability of good actions via gradient update

POLICY GRADIENTS - HISTORY

Policy Gradient Methods for Reinforcement Learning with Function Approximation

Richard S. Sutton, David McAllester, Satinder Singh, Yishay Mansour
AT&T Labs – Research, 180 Park Avenue, Florham Park, NJ 07932

Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning

Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems

ANDREW G. BARTO, MEMBER, IEEE, RICHARD S. SUTTON, AND CHARLES W. ANDERSON

POLICY GRADIENTS - HISTORY

Policy Gradients, REINFORCE, ACTOR-CRITIC 논문 리뷰

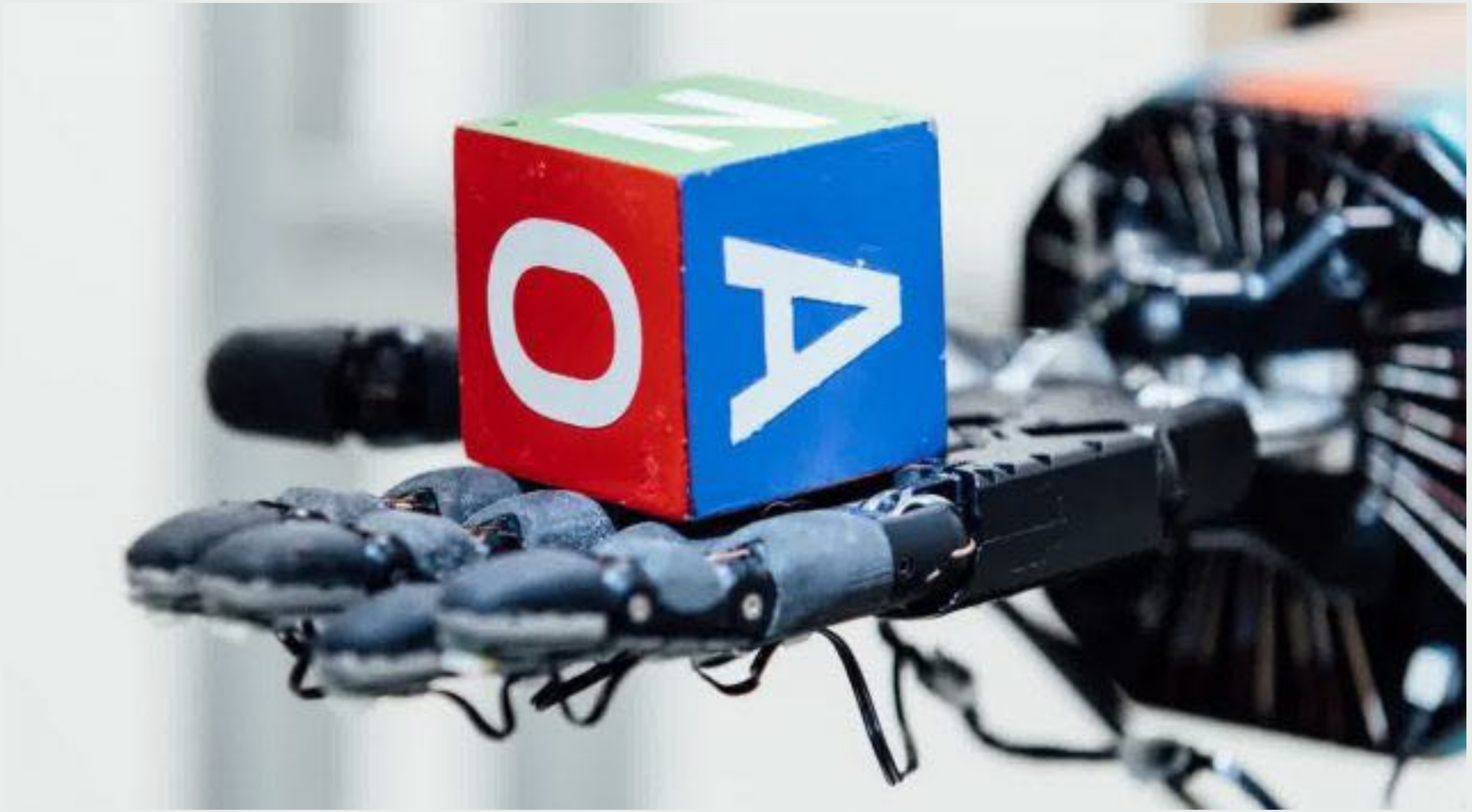
https://wonseokjung.github.io//reinforcementlearning/update/RL-PG_RE_AC/

Proximal Policy Optimization Algorithms

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov
OpenAI
{joschu, filip, prafulla, alec, oleg}@openai.com

ALPHAGO, DOTA

ROBOTIC MANIPULATION



RL REQUIRES A LOT OF TRAINING TIME

	Chess	Shogi	Go
Mini-batches	700k	700k	700k
Training Time	9h	12h	34h
Training Games	44 million	24 million	21 million
Thinking Time	800 sims 40 ms	800 sims 80 ms	800 sims 200 ms

Table S3: Selected statistics of *AlphaZero* training in Chess, Shogi and Go.

RL REQUIRES A LOT OF TRAINING TIME

	OPENAI 1V1 BOT	OPENAI FIVE
CPUs	60,000 CPU cores on Azure	128,000 preemptible CPU cores on GCP
GPUs	256 K80 GPUs on Azure	256 P100 GPUs on GCP
Experience collected	~300 years per day	~180 years per day (~900 years per day counting each hero separately)

PRIOR KNOWLEDGE

from life history

and evolutionary history

HOW WE CAN ALLOW OUT A.I SYSTEM MAKE TO USE PRIOR KNOWLEDGE?

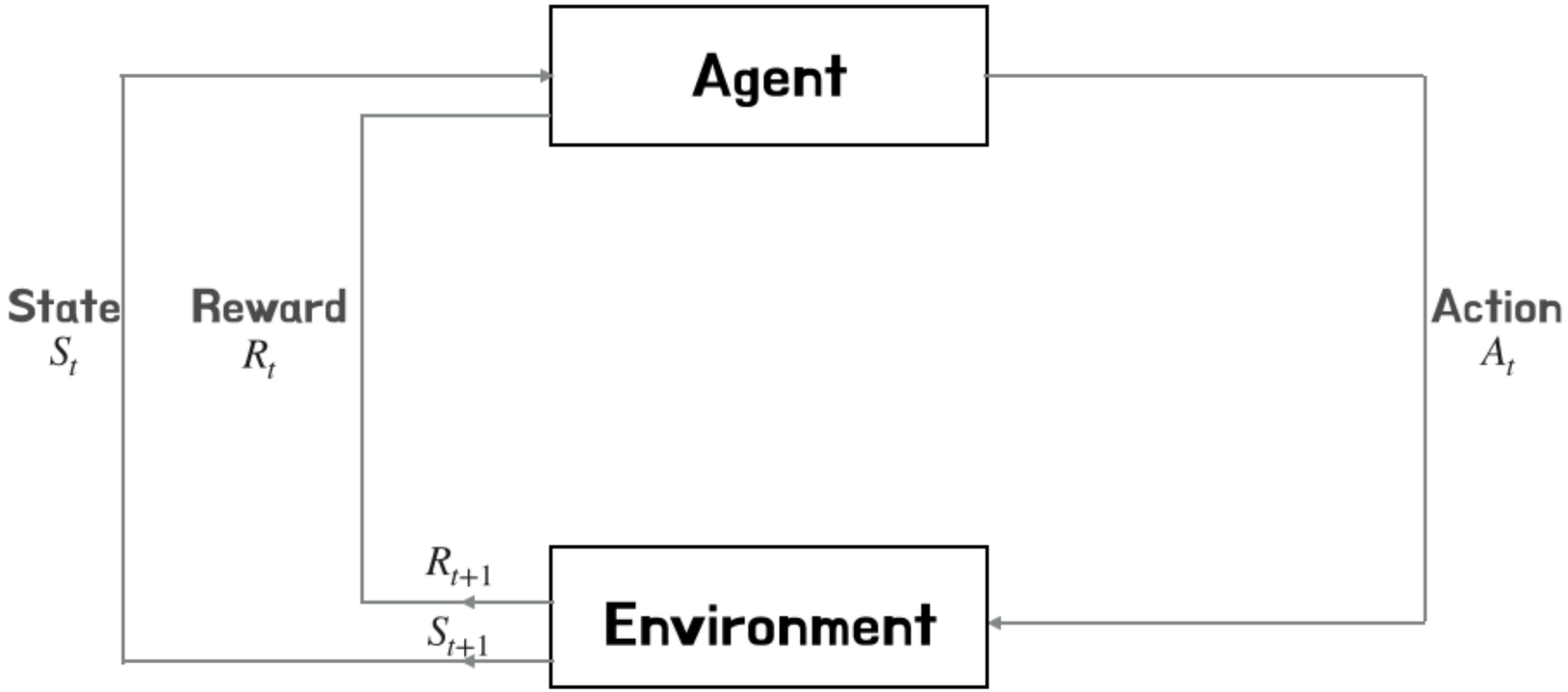
https://ubisafe.org/explore/demeanure-clipart-prior-knowledge/

Meta Reinforcement Learning

https://ubisafe.org/explore/demeanure-clipart-prior-knowledge/

META REINFORCEMENT LEARNING

Sing Task



https://ubisafe.org/explore/demeanure-clipart-prior-knowledge/

SINGLE R.L : MAZE NAVIGATION

Sing Task :

Learn to navigate from start to goal as fast as possible in a single maze

META R.L : MAZE NAVIGATION

"Meta-RL" task: learn to navigate from start to goal as fast as possible
K times in a random maze

META RL IS A SPECIAL CASE OF NORMAL RL

Define new RL task in terms of old task

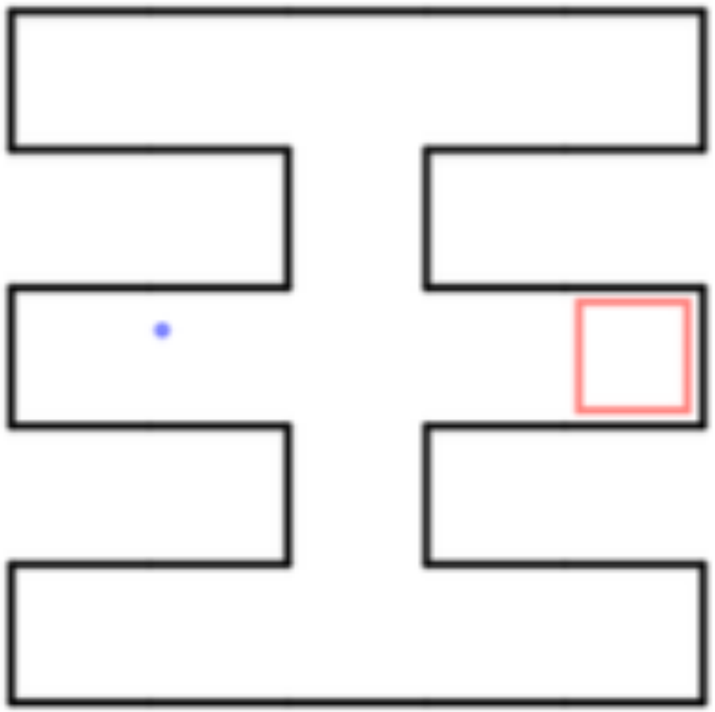
- First timestep: agent is placed in random task / world
- New observation = (old observation, reward, "done" signal)
- New-task episode = K old-task episodes

RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning

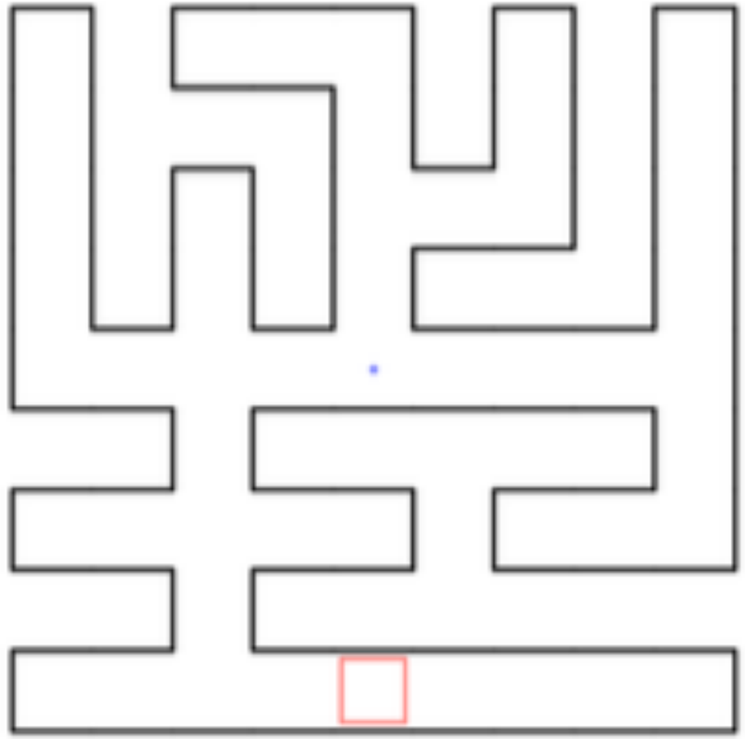
Yan Duan[†‡], **John Schulman**[†‡], **Xi Chen**[†‡], **Peter L. Bartlett**[†], **Ilya Sutskever**[‡], **Pieter Abbeel**[†‡]
[†] UC Berkeley, Department of Electrical Engineering and Computer Science
[‡] OpenAI
{rocky,joschu,peter}@openai.com, peter@berkeley.edu, {ilyasu,pieter}@openai.com

META-RL IN MAZE

(a) Sample observation



(b) Layout of the 5×5 maze in (a)



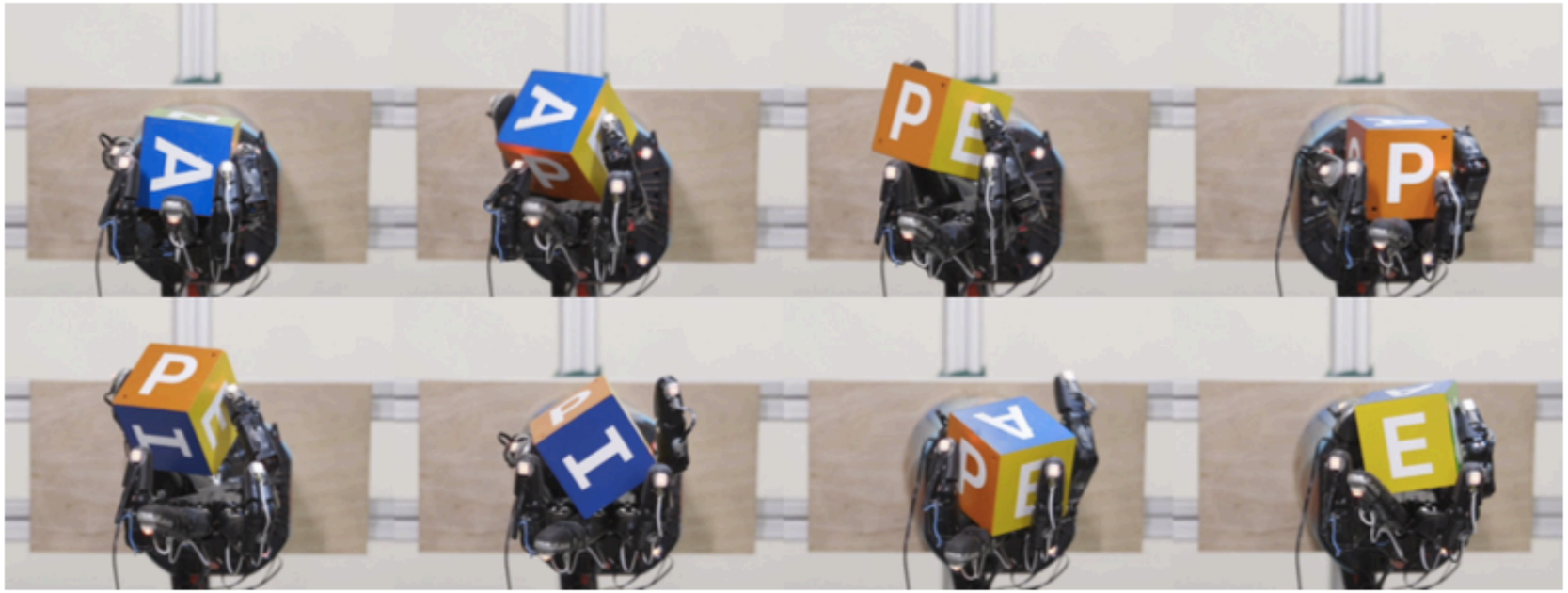
(c) Layout of a 9×9 maze

Figure 4: Visual navigation. The target block is shown in red, and occupies an entire grid in the maze layout.

LEARNING DEXTEROUS IN HAND MANIPULATION

Learning Dexterous In-Hand Manipulation

OpenAI*



META-RL IN ROBOT

META RL : LIMITATION OF APPROACHES DESCRIBE PREVIOUSLY

Meta RL: Limitations of Approaches Described Previously

Infinite data: can randomly sample tasks; assume distribution covers the one you care about

- Doesn't emphasize generalization—performance given finite training set

All "learning" is performed through RNN state updates: might be a poor inductive bias for what learning algorithm should look like

- All meta-learning results so far use short horizons, 1000s of timesteps max
- RL algorithms (policy gradients, Q-learning) find better solutions after longer training periods

META RL : CHANGES TO PROBLEM FORMULATION

Finite set of training tasks and a test set of tasks

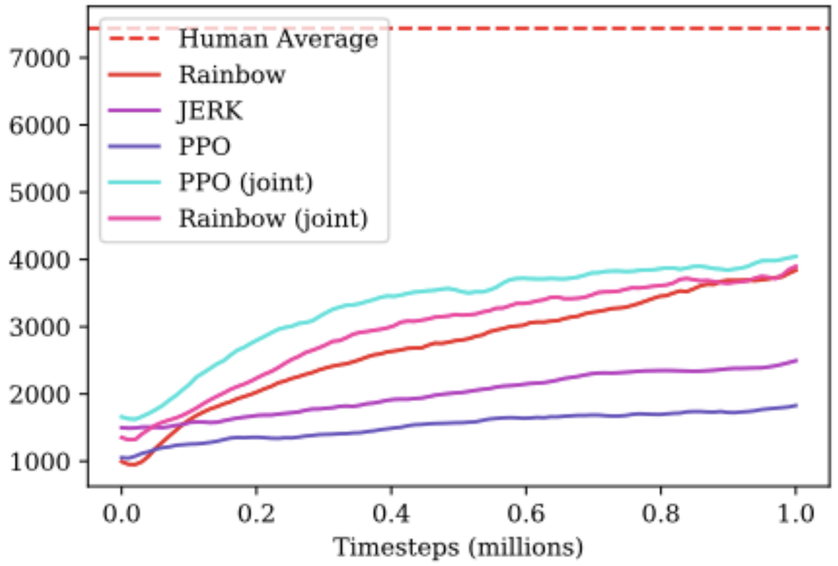
GYM RETRO

REINFORCEMENT LEARNING

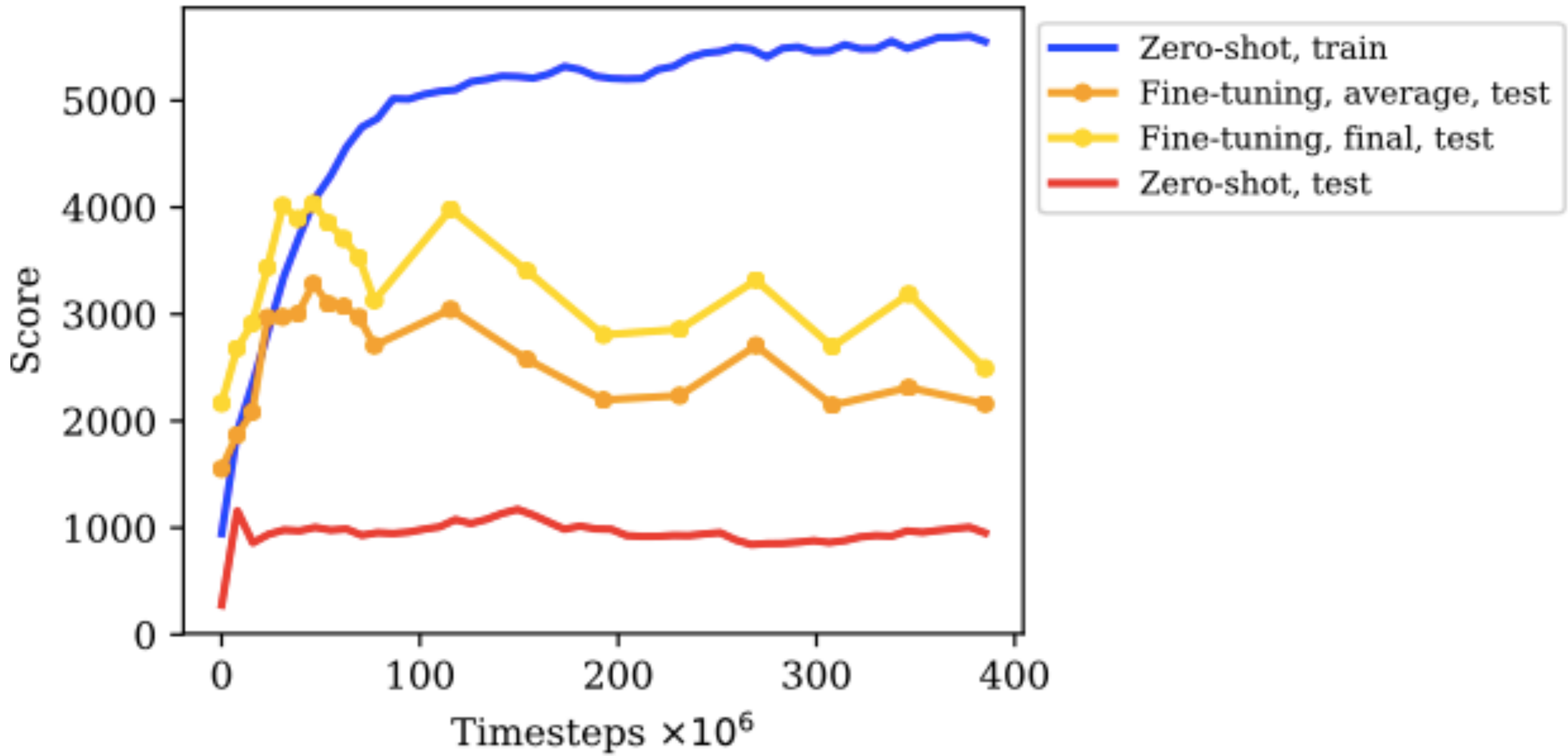
ALGORITHMS

Table 1: Aggregate test scores for each of the baseline algorithms.

Algorithm	Score
Rainbow	2748.6 ± 102.2
JERK	1904.0 ± 21.9
PPO	1488.8 ± 42.8
PPO (joint)	3127.9 ± 116.9
Rainbow (joint)	2969.2 ± 170.2
Human	7438.2 ± 624.2



PPO(JOINT) + FINE TUNING



RETRO CONTEST

contest.openai.com

April 5 to June 5, 2018

Hired level designers to create 11 custom levels

Also created 5 low-quality custom levels for leaderboard

Registration numbers:

923 teams registered, 229 submitted solutions

IMPROVEMENT

- Improve performance on larger retro benchmark
  - Exploration
  - Unsupervised learning
  - Hierarchy
- Improve RNN-based meta-learning
  - Better dealing with long time horizon: memory & credit assignment
  - Better architectures

MODULABS CTRL

#27	Modulabs	4293.71

감사합니다.

Thank you

Github:
https://github.com/wonseokjung

Facebook:
https://www.facebook.com/ws.jung.798

Blog:
https://wonseokjung.github.io/